# Table of contents

1 Foreword ................................................................................................................................. 7
   1.1 Notes on the documentation ............................................................................................... 7
   1.2 Safety instructions ............................................................................................................... 8
   1.3 Notes on information security ........................................................................................... 9

2 Overview ..................................................................................................................................... 10
   2.1 System components .......................................................................................................... 10
   2.2 Licensing model ................................................................................................................ 12

3 Installation .................................................................................................................................. 14
   3.1 System requirements ........................................................................................................ 14
   3.2 Version overview ............................................................................................................. 15
   3.3 Setup instructions ............................................................................................................. 16
   3.4 Licensing .......................................................................................................................... 22

4 First steps ................................................................................................................................... 25
   4.1 Build Project ..................................................................................................................... 25
      4.1.1 Creating a TwinCAT project ...................................................................................... 25
      4.1.2 Choose Target System ............................................................................................. 27
      4.1.3 System configuration ................................................................................................ 29
   4.2 Setting up a Vision device ................................................................................................ 31
      4.2.1 Displaying a Vision node and creating an application ............................................. 31
      4.2.2 Creating a File Source Control ............................................................................... 33
      4.2.3 Creating a GigE Vision camera .............................................................................. 36
   4.3 Creating a PLC project ..................................................................................................... 43
      4.3.1 Creating a PLC project ............................................................................................. 43
      4.3.2 Integrating a PLC library .......................................................................................... 44
      4.3.3 Writing a PLC program ............................................................................................ 46
      4.3.4 Initializing a function block ..................................................................................... 47
   4.4 Run ..................................................................................................................................... 48
      4.4.1 Activate TwinCAT project and start PLC project .................................................. 48
      4.4.2 Image display in ADS Image Watch ......................................................................... 49

5 Development environment ........................................................................................................... 51
   5.1 System configuration ......................................................................................................... 51
      5.1.1 Router memory ........................................................................................................... 52
      5.1.2 Stack size ................................................................................................................... 53
      5.1.3 Network adapter ....................................................................................................... 54
      5.1.4 CPU cores and tasks ............................................................................................... 58
      5.1.5 Job tasks ..................................................................................................................... 61
   5.2 VISION node ...................................................................................................................... 62
      5.2.1 Service Configuration ............................................................................................... 63
      5.2.2 Logging ...................................................................................................................... 64
   5.3 Application node ................................................................................................................ 65
   5.4 GigE Vision Camera objects ................................................................................................. 66
      5.4.1 General ....................................................................................................................... 68
      5.4.2 Configuration Assistant ............................................................................................ 71

TF7000 - TF7300 Version: 1.3
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.4.3</td>
<td>Record/Playback</td>
<td>97</td>
</tr>
<tr>
<td>5.4.4</td>
<td>Camera Calibration</td>
<td>103</td>
</tr>
<tr>
<td>5.4.5</td>
<td>GigE Vision</td>
<td>116</td>
</tr>
<tr>
<td>5.4.6</td>
<td>TcCOM objects</td>
<td>117</td>
</tr>
<tr>
<td>5.5</td>
<td>File Source objects</td>
<td>119</td>
</tr>
<tr>
<td>5.5.1</td>
<td>File Source Control</td>
<td>120</td>
</tr>
<tr>
<td>5.5.2</td>
<td>TcCOM objects</td>
<td>124</td>
</tr>
<tr>
<td>5.6</td>
<td>Vision Job Pool</td>
<td>125</td>
</tr>
<tr>
<td>5.7</td>
<td>ADS Image Watch</td>
<td>125</td>
</tr>
<tr>
<td>5.8</td>
<td>Exporting/Importing nodes</td>
<td>129</td>
</tr>
<tr>
<td>6</td>
<td>API reference</td>
<td>131</td>
</tr>
<tr>
<td>6.1</td>
<td>Software concept</td>
<td>131</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Interface Pointers</td>
<td>131</td>
</tr>
<tr>
<td>6.1.2</td>
<td>HRESULT</td>
<td>135</td>
</tr>
<tr>
<td>6.1.3</td>
<td>Watchdogs</td>
<td>137</td>
</tr>
<tr>
<td>6.1.4</td>
<td>Image processing</td>
<td>140</td>
</tr>
<tr>
<td>6.2</td>
<td>Data Types</td>
<td>150</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Aliases</td>
<td>151</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Constants</td>
<td>151</td>
</tr>
<tr>
<td>6.2.3</td>
<td>Arrays</td>
<td>153</td>
</tr>
<tr>
<td>6.2.4</td>
<td>Enums</td>
<td>153</td>
</tr>
<tr>
<td>6.2.5</td>
<td>Structs</td>
<td>203</td>
</tr>
<tr>
<td>6.3</td>
<td>Interfaces</td>
<td>223</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Containers</td>
<td>223</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Images</td>
<td>352</td>
</tr>
<tr>
<td>6.3.3</td>
<td>Machine Learning</td>
<td>390</td>
</tr>
<tr>
<td>6.3.4</td>
<td>Miscellaneous</td>
<td>394</td>
</tr>
<tr>
<td>6.4</td>
<td>Functions</td>
<td>401</td>
</tr>
<tr>
<td>6.4.1</td>
<td>License Overview</td>
<td>402</td>
</tr>
<tr>
<td>6.4.2</td>
<td>Algebraic Container Operations</td>
<td>427</td>
</tr>
<tr>
<td>6.4.3</td>
<td>Algebraic Image Operations</td>
<td>467</td>
</tr>
<tr>
<td>6.4.4</td>
<td>Basic Container Operations</td>
<td>510</td>
</tr>
<tr>
<td>6.4.5</td>
<td>Basic Image Operations</td>
<td>713</td>
</tr>
<tr>
<td>6.4.6</td>
<td>Code Reading</td>
<td>753</td>
</tr>
<tr>
<td>6.4.7</td>
<td>Container Statistics</td>
<td>770</td>
</tr>
<tr>
<td>6.4.8</td>
<td>Contour Analysis</td>
<td>852</td>
</tr>
<tr>
<td>6.4.9</td>
<td>Control Functions</td>
<td>878</td>
</tr>
<tr>
<td>6.4.10</td>
<td>Drawing</td>
<td>883</td>
</tr>
<tr>
<td>6.4.11</td>
<td>Fourier Analysis</td>
<td>927</td>
</tr>
<tr>
<td>6.4.12</td>
<td>Geometric and Coordinate Transformations</td>
<td>942</td>
</tr>
<tr>
<td>6.4.13</td>
<td>Image Analysis</td>
<td>1026</td>
</tr>
<tr>
<td>6.4.14</td>
<td>Image Color and Contrast Processing</td>
<td>1078</td>
</tr>
<tr>
<td>6.4.15</td>
<td>Image Filtering</td>
<td>1111</td>
</tr>
<tr>
<td>6.4.16</td>
<td>Image Segmentation</td>
<td>1161</td>
</tr>
<tr>
<td>6.4.17</td>
<td>Keypoint Features</td>
<td>1169</td>
</tr>
<tr>
<td>6.4.18</td>
<td>Measurement</td>
<td>1215</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td>------</td>
</tr>
<tr>
<td>6.5</td>
<td>Function Blocks</td>
<td>1281</td>
</tr>
<tr>
<td>6.5.1</td>
<td>License Overview</td>
<td>1281</td>
</tr>
<tr>
<td>6.5.2</td>
<td>Camera Register Access</td>
<td>1282</td>
</tr>
<tr>
<td>6.5.3</td>
<td>File Access</td>
<td>1298</td>
</tr>
<tr>
<td>6.5.4</td>
<td>Image Acquisition</td>
<td>1314</td>
</tr>
<tr>
<td>6.5.5</td>
<td>Image Processing</td>
<td>1353</td>
</tr>
<tr>
<td>7</td>
<td>TwinCAT HMI packages</td>
<td>1359</td>
</tr>
<tr>
<td>7.1</td>
<td>Server extension</td>
<td>1359</td>
</tr>
<tr>
<td>8</td>
<td>Samples</td>
<td>1362</td>
</tr>
<tr>
<td>8.1</td>
<td>Function samples</td>
<td>1363</td>
</tr>
<tr>
<td>8.1.1</td>
<td>Basic Container Operations</td>
<td>1364</td>
</tr>
<tr>
<td>8.1.2</td>
<td>Basic Image Operations</td>
<td>1366</td>
</tr>
<tr>
<td>8.1.3</td>
<td>Code Reading</td>
<td>1369</td>
</tr>
<tr>
<td>8.1.4</td>
<td>Contour analysis</td>
<td>1380</td>
</tr>
<tr>
<td>8.1.5</td>
<td>Image analysis</td>
<td>1390</td>
</tr>
<tr>
<td>8.1.6</td>
<td>Image Color and Contrast Processing</td>
<td>1410</td>
</tr>
<tr>
<td>8.1.7</td>
<td>Image filtering</td>
<td>1415</td>
</tr>
<tr>
<td>8.1.8</td>
<td>Image segmentation</td>
<td>1420</td>
</tr>
<tr>
<td>8.1.9</td>
<td>Measurement</td>
<td>1423</td>
</tr>
<tr>
<td>8.1.10</td>
<td>Self-written functions</td>
<td>1440</td>
</tr>
<tr>
<td>8.2</td>
<td>Function block samples</td>
<td>1442</td>
</tr>
<tr>
<td>8.2.1</td>
<td>Camera Register Access</td>
<td>1442</td>
</tr>
<tr>
<td>8.2.2</td>
<td>File Access</td>
<td>1444</td>
</tr>
<tr>
<td>8.2.3</td>
<td>Image acquisition</td>
<td>1446</td>
</tr>
<tr>
<td>8.3</td>
<td>Samples of assistants</td>
<td>1448</td>
</tr>
<tr>
<td>8.3.1</td>
<td>Calibration Assistant</td>
<td>1448</td>
</tr>
<tr>
<td>8.4</td>
<td>Camera configuration samples</td>
<td>1455</td>
</tr>
<tr>
<td>8.4.1</td>
<td>Region of Interest (ROI)</td>
<td>1455</td>
</tr>
<tr>
<td>8.4.2</td>
<td>Binning</td>
<td>1456</td>
</tr>
<tr>
<td>8.4.3</td>
<td>Image acquisition and trigger</td>
<td>1459</td>
</tr>
<tr>
<td>9</td>
<td>Appendix</td>
<td>1466</td>
</tr>
<tr>
<td>9.1</td>
<td>FAQ</td>
<td>1466</td>
</tr>
<tr>
<td>9.2</td>
<td>Important paths</td>
<td>1467</td>
</tr>
<tr>
<td>9.3</td>
<td>Troubleshooting</td>
<td>1468</td>
</tr>
<tr>
<td>9.3.1</td>
<td>TwinCAT system start</td>
<td>1468</td>
</tr>
<tr>
<td>9.3.2</td>
<td>PLC runtime environment</td>
<td>1472</td>
</tr>
<tr>
<td>9.3.3</td>
<td>Assistants</td>
<td>1479</td>
</tr>
<tr>
<td>9.3.4</td>
<td>TwinCAT HMI packages</td>
<td>1483</td>
</tr>
<tr>
<td>9.3.5</td>
<td>Camera communication</td>
<td>1484</td>
</tr>
<tr>
<td>9.4</td>
<td>ADS Return Codes</td>
<td>1486</td>
</tr>
<tr>
<td>9.5</td>
<td>Support and Service</td>
<td>1491</td>
</tr>
<tr>
<td>9.6</td>
<td></td>
<td>1492</td>
</tr>
</tbody>
</table>
1 Foreword

1.1 Notes on the documentation

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

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

Disclaimer

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

Trademarks

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

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:


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

Copyright

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

Safety regulations

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

Exclusion of liability

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

Personnel qualification

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

Description of symbols

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

⚠️ **DANGER**

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

⚠️ **WARNING**

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

⚠️ **CAUTION**

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

**NOTE**

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

● **Tip or pointer**

This symbol indicates information that contributes to better understanding.
1.3 Notes on information security

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

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

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

To stay informed about information security for Beckhoff products, subscribe to the RSS feed at https://www.beckhoff.com/secinfo.
TwinCAT Vision is a TwinCAT 3 function for industrial image processing. Tasks such as the detection, identification or measurement of objects can be handled directly in the PLC in real-time. By integrating image processing into the TwinCAT platform, highly synchronized control applications and extremely short response times can be achieved. Application development is aided by the large number of existing TwinCAT Engineering tools.

Image processing sequences are programmed in the IEC61131-3 languages using the Tc3_Vision library directly in the PLC code. Intermediate results can be checked directly using an image display tool.

TwinCAT Vision is real-time capable because the image processing algorithms are executed synchronously with the controller directly in the TwinCAT 3 runtime environment. In addition, suitable algorithms can run automatically in parallel on several cores.

A large number of industrial cameras can be connected via the GigE Vision interface. The familiar TwinCAT development environment is used for configuring and calibrating cameras, and for system setup and debugging.

The individual chapters provide detailed information on the following topics:

- **System components [10]** for understanding the system structure of TwinCAT Vision
- **Installation [14]** for guidance on installation and licensing
- **First steps [25]** for a quick introduction to TwinCAT Vision
- **Development environment [51]** for configuration, calibration and simulation of cameras
- **API reference [131]** (software documentation)
- **Examples [1362]** of some image processing sequences
- **Notes [1468]** on troubleshooting

### 2.1 System components

Like other TwinCAT components, TwinCAT Vision is integrated into the Visual Studio-based TwinCAT development environment and the real-time capable TwinCAT runtime environment. The components of the TwinCAT Vision system are explained below.
Configuration Assistants

- Industrial cameras can be connected via the GigE Vision [116] interface. Assistants [66] are available for configuration, calibration and simulation of the cameras.
- Images can be fed from the file system into the PLC runtime via a File Source [119] element.
- The configuration [51] of the cores and the router memory for TwinCAT Vision must always be carried out in the TwinCAT real-time settings.
- TwinCAT Vision can utilize TwinCAT’s multi-core support to execute certain algorithms in parallel. Corresponding settings are made in the Vision Job Pool [125].
Overview

PLC library
TwinCAT Vision provides a PLC library with a large number of algorithms for programming image processing sequences. It also contains function blocks that serve as interfaces for cameras and the file system. The API reference [131] contains detailed descriptions of all library elements.

Debugging tools
TwinCAT Vision provides the following mechanisms for debugging image processing sequences:

- ADS Image Watch [125] can be used for live visualization of camera images and intermediate results. It transfers images from the PLC runtime to the development environment via ADS and displays them there.
- All TwinCAT Vision functions provide information on their execution status based on ADS Return Codes [1486] (see HRESULT [135]).
- All TwinCAT debugging functions such as breakpoints etc. can be used for programming.

GigE Vision driver
TwinCAT Vision includes a real-time capable GigE Vision [116] driver. It is used in the development environment for camera configuration and in the TwinCAT runtime. It communicates with industrial cameras via UDP/IP, based on the GigE Vision standard. This standard is used to find and configure cameras in the network and to receive image data from the cameras.

TwinCAT Vision service
For some functionalities, TwinCAT Vision has to communicate with components outside the real-time at runtime. For example, a File Source Control loads images from the file system. The TwinCAT Vision Service [63] is a centralized, ADS-based communication interface between the TwinCAT runtime and external Vision components.

TwinCAT HMI Vision Extension
The TwinCAT HMI Vision Extension [1359] is an extension for the TwinCAT HMI that enables transfer of images from the PLC to the TwinCAT HMI. Images from the PLC can be displayed directly in the web-based HMI with HmiImageControl.

2.2 Licensing model
The TF7xxx products for TwinCAT Vision comprise licenses for camera connections and an image processing library. To generate licenses, please follow the corresponding guide [22]. For testing or development with TwinCAT Vision, you can create licenses for a 7-day test version.

For currently available product licenses, please refer to the product website.

TF700x | TC3 GigE Vision Connector
The TC3 GigE Vision Connector enables connection of GigE Vision cameras. The basic TF7000 license is always required and enables connection of two cameras. If required, the capacity can be increased with expansion licenses:

<table>
<thead>
<tr>
<th>License</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF7000</td>
<td>Basic license incl. 2 camera connections (always required)</td>
</tr>
<tr>
<td>TF7001</td>
<td>Extension for 2 additional connections</td>
</tr>
<tr>
<td>TF7002</td>
<td>Extension for 4 additional connections</td>
</tr>
<tr>
<td>TF7003</td>
<td>Extension for 8 additional connections</td>
</tr>
</tbody>
</table>

The extension licenses can be combined so that a maximum of 2+2+4+8=16 cameras can be connected.
For example, the following licenses are required for 11 camera connections:

- TF7000 base (including 2 camera connections)
- TF7001 2 cameras
- TF7003 8 cameras

→ 12 cameras can now be connected in total.

**TF7100-TF7300 | TC3 Vision algorithms**

The TF7100-TF7300 product licenses provide image processing functions in the "Tc3_Vision" PLC library. The basic TF7100 license is required for any use of the library. It provides access to basic algorithms such as filter operations, format conversions and contour tracing. This already allows many applications to be implemented. The extension licenses supplement the range of functions as described in the table. The assignment of functions to licenses can be found in the API reference [402].

<table>
<thead>
<tr>
<th>License</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF7100</td>
<td>Basic license for the &quot;Tc3_Vision&quot; library. Contains algorithms for the solving of image processing tasks, such as algebraic operations, filters, Fourier analysis, color image processing, segmentation, contour and blob analysis, results display and for the reading and writing of camera parameters.</td>
</tr>
<tr>
<td>TF7200</td>
<td>Extension for 2D matching: Objects can be found and compared on the basis of learned references, contours, feature points or other properties (Template Matching / KeyPoint Detection and Descriptor Matching). A common application is the classification of objects.</td>
</tr>
<tr>
<td>TF7250</td>
<td>Extension for Code Reading: Detection and reading of 1D and 2D codes.</td>
</tr>
</tbody>
</table>
| TF7300    | Extension for 2D metrology: Tools for optical measurement of geometric object properties (distances, radii etc.).
  - Edge localization on the measured images with subpixel accuracy
  - Compensation of optical distortions (requires calibration of the camera!)
  - Transformation between pixel coordinates and real-world coordinates (requires calibration of the camera!) |

**Automatic determination of the required Vision library licenses**

The required licenses are only determined when compiling the respective TwinCAT project. At the same time, a check is carried out to see which function blocks are declared and which functions are used in the code.

Therefore, please note that required licenses may be need to be added or omitted in the case of project changes (at short notice). If new licenses are added, the change must be loaded by means of an **Activate Configuration**; an online change is not possible in this case.
3 Installation

This chapter contains the system requirements [14] for TwinCAT Vision, the version overview [15] and instructions for installation [16] and licensing [22].

3.1 System requirements

IPC hardware

<table>
<thead>
<tr>
<th>Technical data</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT 3 Platform-Level</td>
<td>Minimum P50 Performance Plus, e.g. Intel 4-core Atom CPU</td>
</tr>
<tr>
<td>Network adapter</td>
<td>TwinCAT real-time Ethernet-compatible Gigabit network cards (for connecting GigE Vision cameras)</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong> At least one network port should be available exclusively for cameras. It is recommended to use a separate network port for each camera.</td>
</tr>
<tr>
<td>Main memory</td>
<td>Minimum 4 GB recommended for pure runtime systems</td>
</tr>
<tr>
<td></td>
<td>Minimum 8 GB recommended for development systems</td>
</tr>
<tr>
<td></td>
<td>Minimum 8 GB recommended for the use of record/playback [97] streams on runtime systems</td>
</tr>
<tr>
<td>Hard disk</td>
<td>SSD hard disks with high reading/writing rates are recommended, especially if a lot of data is to be exchanged between PLC and hard disk.</td>
</tr>
</tbody>
</table>

Camera

<table>
<thead>
<tr>
<th>Technical data</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>Must support GigE Vision [116]. (marked with GigE Vision logo)</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong> The TwinCAT GigE Vision Connector is certified according to the GigE Vision Standard version 2.0. However, this does not mean that all camera or GigE Vision features of the version are supported.</td>
</tr>
</tbody>
</table>

Software

<table>
<thead>
<tr>
<th>Technical data</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating system</td>
<td>Windows 7 or 10 (64-bit only!)</td>
</tr>
<tr>
<td>Supported Visual Studio</td>
<td>2017 (Community, Professional)</td>
</tr>
<tr>
<td></td>
<td>2019 (Community, Professional)</td>
</tr>
<tr>
<td></td>
<td>TwinCAT XAE Shell (TcXaeShell)</td>
</tr>
<tr>
<td>TwinCAT</td>
<td>Minimum version 3.1.4024.17</td>
</tr>
<tr>
<td>.NET Framework</td>
<td>Minimal version 4.6.1 (only for the development environment [51])</td>
</tr>
</tbody>
</table>
3.2 Version overview

<table>
<thead>
<tr>
<th>Setup</th>
<th>TcCOM objects</th>
<th>Tc3_Vision library</th>
<th>TwinCAT Vision service</th>
<th>Engineering Extensions</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.0.2.13</td>
<td>4.0.2.13</td>
<td>4.0.2.13</td>
<td>4.0.2.13</td>
<td>4.0.2.13</td>
<td>Repository files are deleted with the installation of a newer setup.</td>
</tr>
<tr>
<td>4.0.1.3</td>
<td>4.0.1.0</td>
<td>4.0.1.0</td>
<td>4.0.1.0</td>
<td>4.0.1.0</td>
<td>Note If you want to keep this version, save the &quot;%TWINCAT3DIR% \3.1\Repository\Beckhoff Automation GmbH&quot; folder before installing a new setup and copy it back after the installation.</td>
</tr>
</tbody>
</table>

**NOTE**

Compatibility

If you have several TwinCAT Vision versions installed on one system or if the development and runtime environment are on different systems, you must ensure that you use all components with the same version.

The TwinCAT Vision Service and the Engineering Extensions are only available in the latest version. Therefore, these components determine the TcCOM and library version to be used. Mixing versions is not permitted and can lead to malfunctions.

Hints for the selection of a version

If you want to use TwinCAT Vision with a specific version, all TcCOM objects of the project must be updated to the corresponding version. An exception is the Ads Communicator CONFIGONLY. This module is only used internally in the Configuration Assistant and must always have the latest version that matches the extension. Therefore, this module automatically sets the latest version when loading a project.

To check or change the TcCOM versions:

1. Open the window SYSTEM > TcCOM Objects > Project Objects.
2. To change, select the desired version for all objects.
3. Right-click on one of the object IDs in the first column.
4. Select Reload TMI/TMC Description(s) with changed version.

Furthermore, the same version of the Tc3_Vision library, corresponding to the selected TcCOM object version, must be used in the PLC project. Therefore check in the properties of the Tc3_Vision library which version is effectively used or adjust it accordingly.
The current version number of the TwinCAT Vision Service and thus also of the Engineering Extensions can be found in the Service Configuration \[63\] under the Vision node.

### Online Change after update

If you update TwinCAT Vision on a development system and thus make an Online Change on a target system with an older version, this can lead to version conflicts with interface pointers (images and containers).

### Deleting older versions

To delete older versions, two steps are necessary. Firstly, you have to delete the corresponding version folders in the directories Tc3_Vision, TcIoGigEVision and TcVision under "%TWINCAT3DIR% \3.1\Repository\Beckhoff Automation GmbH". To remove the corresponding PLC library version, you have to uninstall it via the library repository dialog.

### 3.3 Setup instructions

The installation of TwinCAT Vision on a Windows operating system consists of the following steps:

1. Make sure that TwinCAT is in Config mode and that no Visual Studio instance or TcXaeShell is open.
2. Run the installation file TF7xxx-Vision.exe as administrator. To do this, right-click the file and select Run As Admin. Confirm by clicking Yes.
3. Select the installation language. Then click OK.
4. Wait a moment while the installation is being prepared, then click Next.

5. Make sure that you agree to the license agreement by checking the option I accept the terms of the license agreement. You can print the license agreement, if required. Then click Next.
6. Enter your user name and company as user information. Then click **Next**.

![Customer Information](Image)

7. Choose between the two setup types **Complete** or **Custom**. Then click **Next**.

![Setup Type](Image)

- A Complete setup installs all TwinCAT Vision components and requires the most memory space. This option is recommended for XAE systems. Since only the Vision Service can be installed on pure runtime systems and the other options are not available for selection, this option is also recommended here.
- A custom setup lets you choose in the next step which TwinCAT Vision components you want to install.

8. For each of the following TwinCAT Vision components, select whether you want to install them or not. In addition, select at least one TwinCAT 3 version in which the selected components are to be installed. Then click **Next**.
- **TC3 Vision**: Required for developing TwinCAT Vision applications and should be installed on XAE systems.
- **TC3 GigE Vision Connector**: Required for developing with GigE Vision devices and should be installed on XAE systems.
- **TC3 Vision Service XAR**: Contains the TwinCAT Vision Service and must be installed on target systems of a TwinCAT Vision application.

**Explanation**

The Vision Service must be installed on the target system. The components for TC3 Vision and TC3 GigE Vision Connector, conversely, are always copied automatically from the development system to the target system when activating the TwinCAT project. Therefore, they do not necessarily have to be installed on the target system.
9. Please note that during the following installation the TwinCAT system service is stopped and restarted in Config mode. Continue with the installation only if a running PLC system can be stopped. In this case, click **Next**.

10. Make sure that you agree with the selected installation settings. Then click **Next**.
11. Wait while the selected TwinCAT Vision components are installed.

12. A dialog may appear showing a list of applications that have to be closed for a complete installation. Choose whether the applications should be automatically closed or whether you would like to restart the PC after the installation instead.

⇒ The installation is now complete. Optionally, you can choose to display the log file of the installation. Click Finish.

TwinCAT Vision must be licensed before it can be used after a successful installation. The procedure is described in the chapter on Licensing [22].

Information about versions and updating existing projects can be found at Version overview [15].
If the installation fails, you have the following options:

- Solve the problem using troubleshooting.
- Contact customer support and provide the installation log. The installation log TwinCAT-Vision.log is located in the following ZIP folder in the path: C:\Users\<username>\BeckhoffSetupLogFiles.zip

### 3.4 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).

**Licensing the full version of a TwinCAT 3 Function**

A description of the procedure to license a full version can be found in the Beckhoff Information System in the documentation "TwinCAT 3 Licensing".

**Licensing the 7-day test version of a TwinCAT 3 Function**

A 7-day test version cannot be enabled for a TwinCAT 3 license dongle.

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. "TF4100 TC3 Controller Toolbox").

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 is displayed and confirm the entry.

9. Confirm the subsequent dialog, which indicates the successful activation.
   
   In the tabular overview of licenses, the license status now indicates the expiry date of the license.
10. Restart the TwinCAT system.
   - The 7-day trial version is enabled.
4  First steps

The following guide provides an introduction to TwinCAT Vision (after successful installation [14]).

1. Build Project [25]
   - Creating a TwinCAT project [25]
   - Choose Target System [27]
   - System configuration [29]
2. Displaying a Setting up a Vision device [31] and creating a Vision device
   - Displaying a Vision node and creating an application [31]
   - Creating a File Source Control [33]
   - Creating a GigE Vision camera [36]
3. Creating a PLC project [43]
   - Creating a PLC project [43]
   - Integrating a PLC library [44]
   - Writing a PLC program [46]
   - Initializing a function block [47]
4. Run [48]
   - Activate TwinCAT project and start PLC project [48]
   - Image display in ADS Image Watch [49]
   - Download sample project: https://github.com/Beckhoff/TF7xxx_Samples

More detailed information can be found at Development environment [51], API reference [131] and Samples [1362].

4.1  Build Project

Video example for setting up a system for TwinCAT Vision:

Video:: https://infosys.beckhoff.com/content/1033/tf7xxx_tc3_vision/Resources/mp4/5906036107.mp4

4.1.1  Creating a TwinCAT project

TwinCAT Vision is fully integrated in the TwinCAT system. A standard TwinCAT project is therefore the starting point for every TwinCAT Vision application:

1. Open Visual Studio with integrated TwinCAT.
2. Open the **New/Project** dialog.

3. Select name and location.
A TwinCAT project is created.

Next step

Choose Target System [27]

4.1.2 Choose Target System

TwinCAT distinguishes between development and runtime systems. If your development system is not your runtime system, create a route to the runtime system and select it. This can be done by following the steps below, for example:

1. Click Solution Explorer > SYSTEM > General > Choose Target...

The Choose Target System dialog opens, listing all target systems to which routes exist from the development system.
2. If the target system is not yet listed, that is there is no route to the system, click **Search**.

The Add Route Dialog opens, where you can add more routes.

3. In the **Add Route Dialog** click **Broadcast Search** to search for TwinCAT systems in the network. Alternatively, search for the device name or IP address using **Enter Host Name / IP**. Any systems that are found are then displayed.
4. Select a system from the list and click **Add Route** to create a route to it. The displayed information and setting options are described in detail in the chapter **Add Route Dialog**. The new route now appears in the **Choose Target System** dialog.

5. Select the target system in the **Choose Target System** dialog and confirm with **OK**.

**Next step**

**System configuration** [51]

### 4.1.3 System configuration

The system configuration depends on the target system and is described in detail in chapter **System configuration** [51].

#### Increase router memory

TwinCAT Vision requires dynamic memory allocation when creating an image, for example. This takes place in the router memory [52]. Accordingly, the router memory must be configured with adequate size.

This is done in the TwinCAT project under **SYSTEM > Real-Time > Settings > Router Memory (Mbyte)**.

The configuration must then be activated . If the router memory is reduced, it is necessary to restart the target system.
Selection of a CPU core and a task

In multi-core systems, TwinCAT enables a distinction between Windows cores and isolated cores. With Windows cores, the operating system and the TwinCAT application share the processor time. The real-time portion can be limited to between 10% and 90%. In contrast, isolated cores are fully available to the TwinCAT application. For this reason, the use of isolated cores is recommended for Vision applications. Divided cores can also be used without problems for the initial testing.

The core configuration is carried out in the TwinCAT project tree under SYSTEM > Real-Time > Settings. Detailed configuration instructions can be found in the chapter on CPU cores and tasks. This chapter also describes how to change the cycle time of the executing task.

The configuration must then be activated. In addition, the target system must be restarted when redefining divided and isolated cores.

Installing a real-time Ethernet driver

To connect GigE Vision cameras to TwinCAT Vision, the TwinCAT RT Ethernet driver must be installed on the network adapters used. The driver ensures that TwinCAT can use the network adapter during runtime. Details can be found in the chapter on Network adapter.

In the development environment, an installation assistant can be called up via TWINCAT > Show Realtime Ethernet Compatible Devices... On the target system the assistant can be started via the file C:\TwinCAT\3.1\System\TcRteInstall.exe. The RT Ethernet driver can be installed for the network adapters listed under Compatible devices in the dialog that opens.
Make sure that all network adapters you want to connect a camera are now in the list of **Installed and ready to use devices (realtime capable)**.

**Next step**

**Displaying a Vision node and creating an application**

4.2  **Setting up a Vision device**

Video example for working with File Source Control:

Video:: [https://infosys.beckhoff.com/content/1033/tf7xxx_tc3_vision/Resources/mp4/5931182859.mp4](https://infosys.beckhoff.com/content/1033/tf7xxx_tc3_vision/Resources/mp4/5931182859.mp4)

Video example for creating a camera instance:

Video:: [https://infosys.beckhoff.com/content/1033/tf7xxx_tc3_vision/Resources/mp4/5931185163.mp4](https://infosys.beckhoff.com/content/1033/tf7xxx_tc3_vision/Resources/mp4/5931185163.mp4)

4.2.1  **Displaying a Vision node and creating an application**

Within the TwinCAT project there is a Vision node for the Vision configuration, which is hidden by default.
1. Right-click the TwinCAT project in the Solution Explorer > Show Hidden Configurations > Show VISION Configuration.

The Vision node is now visible.

2. Within the Vision node, folders (also referred to as applications) are used to group the image processing devices.

Right-click on Vision Node > Add New Item.
3. Select a name and confirm with OK.

![Insert Folder](image)

- The folder is created

![Solution Tree](image)

In addition, you can find further information here:

- VISION node [62]
- Application node [65]

**Next step**

In the next step, a Vision device is created which can be used to load images into the PLC. This can be either a real GigE Vision camera or a File Source Control for loading existing image files:

Creating a GigE Vision camera [36] or Creating a File Source Control [33]

### 4.2.2 Creating a File Source Control

The creation of a File Source [119] object is described in the following steps. This can be used to load stored images into the PLC. To do this, the target system must be in Config Mode. Click on the symbol in the TwinCAT XAE Base toolbar or on Restart TwinCAT (Config Mode) in the Visual Studio menu under TWINCAT.

Alternatively, you can Creating a GigE Vision camera [36].

Proceed as follows to create the File Source object:

1. Click an application under the VISION node you created earlier.
2. Right-click and select **Add new item...**

   - Right-click and select **Add new item...**
   - Solution 'TcVision_FirstSteps' (1 project)
   - TcVision_FirstSteps
     - SYSTEM
     - MOTION
     - PLC
     - SAFETY
     - C++
     - VISION
   - Vision Job Pool
   - Application1
   - ANALYTICS
   - I/O
   - File Source
   - The File Source Control is now created below the application and is ready to use.

3. Select the device type **File Source**, adjust the name if necessary, click **OK**.

   - The File Source Control is now created below the application and is ready to use.
4. Double-click the **File Source Object** underneath the application node in the project tree. In the window that opens, select the **File Source Control** tab and implement the relevant settings.

5. Click **Read From Target**. The **File Source Control** in the project is compared with any images stored on the selected target system.

6. Add images to the File Source Control via **Add Files...**, **Add Directory...** or by drag and drop. If necessary, you can download sample images with the project download: First steps [25]

7. If necessary, adjust the cycle time in which the File Source Object sends images to the PLC.

   The File Source Object can be used like this:

---

**Next step**

Creating a PLC project [43]
4.2.3 Creating a GigE Vision camera

The creation of a GigE Vision Camera object is described in the following steps. This requires a GigE Vision camera to be connected to the PC and the TwinCAT RT Ethernet adapter to be installed for the corresponding network port, as described in System configuration.

In addition, the target system must be in Config Mode. Click on the symbol in the TwinCAT XAE Base toolbar or on Restart TwinCAT (Config Mode) in the Visual Studio menu under TWINCAT.

Alternatively, you can create a File Source Control.

To create a camera, proceed as follows:

1. Click an application under the Vision node you created earlier.
2. Right-click and select Add new item...

3. Select the device type GigE Vision Camera. Customize the camera name and GigE Vision camera options as required. An Ads communicator is required for recording streams; an image provider is the interface to the corresponding PLC function blocks and is therefore usually always required (see camera details). Click on OK.
4. GigE Vision cameras communicate via UDP/IP. A suitable network device with a corresponding IP stack in the IO node is required for the network port used. Create this by clicking **New...**. 

**Note**: Only one network device with one IP stack may be created for each physical port of a network card.

5. To create a new network device, click **NEW...** in the **Select Network Device** dialog.
6. In the **Device Found At** dialog, select the network port to which the camera is connected and click **OK**. 

   **Note**: Only network ports for which the TwinCAT RT Ethernet driver is installed are displayed.

7. The network device is now created in the IO configuration. Select it in the **Select Network Device** dialog and click **OK**. This creates an IP stack for the device.
8. Select the IP stack and click **OK**.

9. If no IPv4 address has been assigned to the network port yet, answer the question whether you want to assign a fixed IP address.

If Yes, set the IP address in the following dialog.

The IP address and subnet mask are then saved in the TwinCAT IP configuration of the network adapter (see [Network configuration](#55)).
10. In the **Camera Initialization Assistant** click **Discover Devices**. The selected network port is searched for connected cameras.

11. All cameras that are found are displayed. Select the camera you require and click **OK**.
The camera is now created under the application and can be used.

Double-clicking the camera node [66] opens a window in which the connection to the camera can be verified. The General [68] tab provides general information about the camera.
In the Configuration Assistant [71] you can use **Start Acquisition** to view the live image of the camera and make configuration settings on the camera.

See also: Camera configuration samples [1455]
Next step

Creating a PLC project [43]

4.3 Creating a PLC project

Video example for creating a PLC project with TwinCAT Vision:

Video:: https://infosys.beckhoff.com/content/1033/tf7xxx_tc3_vision/Resources/mp4/5977084043.mp4

4.3.1 Creating a PLC project

Like the programming of the general machine control, programming of the image processing also takes place in the PLC. To this end, the TwinCAT project must be extended by a PLC project.

1. Right-click the PLC node in the Solution Explorer and select Add New Item...
2. Select the name and location in the **Add New Item** dialog.

The PLC project was created.

Next step

**Integrating a PLC library** [44]

4.3.2 **Integrating a PLC library**

The Tc3_Vision library contains the data types, functions and function blocks required for image processing.
1. To add them to the PLC project, right-click on References and select Add library...

2. Select the Tc3_Vision library and confirm with OK.
First steps

- The library is now added and can be viewed by double-clicking it.

Next step

Writing a PLC program [p. 46]

4.3.3 Writing a PLC program

A simple PLC program for continuous image acquisition from a camera or a file source includes the following:

Variables

<table>
<thead>
<tr>
<th>PROGRAM MAIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>VAR</td>
</tr>
<tr>
<td>hr           : HRESULT;</td>
</tr>
<tr>
<td>fbCamera     : FB_VN_SimpleCameraControl;</td>
</tr>
<tr>
<td>eState       : ETcVnCameraState;</td>
</tr>
<tr>
<td>ipImageIn    : ITcVnImage;</td>
</tr>
<tr>
<td>ipImageInDisp: ITcVnDisplayableImage;</td>
</tr>
<tr>
<td>nNewImageCounter : UINT;</td>
</tr>
</tbody>
</table>

END_VAR

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hr</td>
<td>HRESULT</td>
<td>All functions and many methods of the Tc3_Vision library return a result code of type HRESULT for error handling. Further information about the concept behind it and the meaning of individual codes can be found under HRESULT [135].</td>
</tr>
<tr>
<td>fbCamera</td>
<td>FB_VN_SimpleCameraControl</td>
<td>Each camera and File Source Control is represented by a function block instance. In the simplest case this is an instance of the FB_VN_SimpleCameraControl [1348], which covers the common properties of camera and File Source Control and can therefore be linked with both device types. Alternatively, FB_VN_GevCameraControl [1324] can be used for cameras and FB_VN_FileSourceControl [1316] for File Source Controls.</td>
</tr>
<tr>
<td>eState</td>
<td>ETcVnCameraState</td>
<td>Each Vision device has a state of type ETcVnCameraState [158].</td>
</tr>
<tr>
<td>ipImageIn</td>
<td>ITcVnImage</td>
<td>Images for processing are managed using ITcVnImage [383] variables. In this sample, access to the camera input image is gained via ipImageIn.</td>
</tr>
<tr>
<td>ipImageInDisp</td>
<td>ITcVnDisplayableImage</td>
<td>Images for display are managed via ITcVnDisplayableImages [383].</td>
</tr>
</tbody>
</table>
Program code with state machine

Each Vision device has an internal state, TCVN_CS_INITIAL being the initial state. From this, the device can be brought into the TCVN_CS_ACQUIRING state using the StartAcquisition method. In this state, images can be captured and transferred. If an image is completely stored in the memory, it can be retrieved via fbCamera.GetCurrentImage(ipImageIn).

To check if an image was received completely and could be retrieved, the IF query queries the HRESULT of the method and whether the pointer to the memory area of the image is not 0. After that, you can start analyzing the image. To check and indicate that an image could be retrieved, the counter variable nNewImageCounter is incremented as an example.

F_VN_TransformIntoDisplayableImage [750] then moves the image pointer to ipImageInDisp to display the image. The image can then no longer be retrieved via ipImageIn. Alternatively, a copy can be created with F_VN_CopyIntoDisplayableImage [721] to display and edit the image.

```plaintext
  eState := fbCamera.GetState();
  IF eState = TCVN_CS_ERROR THEN
    hr := fbCamera.Reset();
  ELSIF eState < TCVN_CS_ACQUIRING THEN
    hr := fbCamera.StartAcquisition();
  ELSIF eState = TCVN_CS_ACQUIRING THEN
    hr := fbCamera.GetCurrentImage(ipImageIn);
      // Check if new Image was received
      IF SUCCEEDED(hr) AND ipImageIn <> 0 THEN
        nNewImageCounter := nNewImageCounter + 1;
        // Place to call vision algorithms
        hr := F_VN_TransformIntoDisplayableImage (ipImageIn, ipImageInDisp, hr);
      END_IF
  END_IF
```

Further information on the state machine can be found in the API chapter Image Acquisition [1314].

Further information

- API reference [131]
- Function blocks for image acquisition [1314]
- Images [140]

Next step

Initializing a function block [47]

4.3.4 Initializing a function block

Following the configuration of a Vision device and the creation of a PLC program, the Vision device must now be linked with appropriate camera variables in the PLC.

The instance fbCamera of the function block FB_VN_SimpleCameraControl [1348] from the PLC program must be linked with the image provider of the Vision device. Please note:
1. Build the PLC program by right-clicking the PLC project and selecting Build. Make sure that no errors occur when doing this. If this is the case, the Symbol Initialization tab appears in the PLC instance and the fbCamera instance below it.

2. Double-click the instance of the PLC project and open the Symbol Initialization tab.

3. Assign the Image Provider of your Vision device to the symbol MAIN.fbCamera.oidITcVNImageProvider via the drop-down in the Value column. This is shown for a camera object in the picture, but it works in exactly the same way for File Source objects.

Next step

Activate TwinCAT project and start PLC project

4.4 Run

Video example for executing a TwinCAT Vision PLC project:

Video: https://infosys.beckhoff.com/content/1033/tf7xxx_tc3_vision/Resources/mp4/5977086347.mp4

4.4.1 Activate TwinCAT project and start PLC project

You can execute the project by performing the following steps:

1. Activate the configuration via or via the Visual Studio menu under TwinCAT > Activate Configuration.
2. For each connected camera the following Init Command dialog appears, unless the Init Commands correspond to the current configuration or the message box was deactivated. For an initial test the Init Commands can be ignored with No.

3. If no licenses have been activated yet on the target system, the following dialog appears. For an initial test you can create a trial license with Yes and enter a security code.

4. You will then be asked whether you want to restart TwinCAT. Confirm with Yes. Alternatively, a restart can be triggered via or via the Visual Studio menu under TwinCAT > Restart TwinCAT System.

5. If the PLC is not set to autostart, start it manually. To do this, log into the PLC via or via the Visual Studio menu under PLC > Login.

6. Then start the PLC via or via the Visual Studio menu under PLC > Start.

Next step

Image display in ADS Image Watch [49]

4.4.2 Image display in ADS Image Watch

Images of type ITcVnDisplayableImage in the PLC can be displayed in ADS Image Watch. You can find the window in the Visual Studio menu TwinCAT > Windows > ADS Image Watch.

To select the image to display, set the following parameters from left to right:

- Target system
- Port
- ADS symbol of the image
First steps

- Scaling of the image

In addition to the image content, the following information can be read in the ADS Image Watch:
  - Histogram of the image
  - Color/intensity values at the current cursor position
  - Pixel coordinates of the current cursor position
  - Image size
5 Development environment

TwinCAT Vision is fully integrated in the TwinCAT 3 development environment. Essentially, TwinCAT Vision extends the TwinCAT system by a further node, below which the Vision components can be created and configured. The following configuration elements are used:

- **The VISION node** [62] provides general configuration options that affect all Vision components on the system. It includes all other configuration elements.

- **The Vision Job Pool** [125] can be used to define job tasks for parallel execution of Vision functions. Each system has a single Vision Job Pool.

- **An application node** [65] encapsulates several Vision devices (GigE Vision cameras and File Sources) to form an applicative unit. The entire configuration of an application can be exported and imported. Each TwinCAT project can have several applications.

- **A GigE Vision camera node** [66] contains the configuration of a physical GigE Vision camera. Each application can have several GigE Vision camera nodes.

- **A File Source node** [119] contains the configuration for an interface between images on the hard disk of the target system and an image acquisition module within the PLC. Each application can have several File Source nodes.

### Communication with system components

GigE Vision camera nodes and File Source nodes each contain TcCOM modules for communication with various system components.

Furthermore, the development environment features an **ADS Image Watch** [125] window element for displaying images from the PLC via ADS. In addition, a general **system configuration** [51] is necessary for the correct functioning of the Vision components.

In order to be able to display images from the PLC live in a TwinCAT HMI, a corresponding **HMI extension** [1359] is available.

5.1 System configuration

The following sections describe the settings specific to TwinCAT Vision, which should be carried out or checked in relation to the system and project.
5.1.1 Router memory

Compared to conventional control systems, image processing applications require a lot of memory, because even a single image can contain several megabytes of data. In TwinCAT Vision, all dynamic data (see Interface Pointer [131]) are allocated in the router memory. For this reason, it is particularly important to configure the size of the router memory accordingly. It can be set in the TwinCAT system configuration under the Real-Time node in the Settings tab:

<table>
<thead>
<tr>
<th>Configured Size</th>
<th>Setting of the router memory required for the current project.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allocated</td>
<td>Router memory currently allocated on the system.</td>
</tr>
<tr>
<td>Available</td>
<td>The part of the router memory currently allocated on the system that is still freely available. (The rest is already used within TwinCAT.)</td>
</tr>
</tbody>
</table>

*Note* If the available memory is not fully restored following a TwinCAT restart, this points to a memory leak. In that case, restart your system and check your program with regard to memory leaks.

The maximum adjustable size of the router memory is currently 1024 MB. The actually required memory size depends among other things on the use of dynamic memory (e.g. with images [140] and containers [143]) in the PLC program.

### NOTE

**Size of the router memory**

When configuring the router memory, it must be noted that the router memory will be subtracted from the RAM and is thus no longer available to the operating system. Therefore we recommended avoiding the maximum size of 1024 MB in systems with a RAM size of less than 4 GB. With external projects, always check the router memory setting before activating the configuration and reduce it if necessary.

**Estimating the required memory size**

The required size of the router memory depends on many factors. A rough estimate for the Vision part of the TwinCAT project can be obtained by calculating the memory dynamically allocated (by Interface Pointers [131]).

The memory space \( M \) (in bytes) of an image can be calculated by incorporating the image size (width \( w \) and height \( h \)), number of channels \( C \) and the number of bytes of the element type: \( M = w \times h \times C \times \text{sizeof(ElementType)} \).

For a container of the type String or Vector, the memory space \( M \) (in bytes) can be calculated through the number of allocated elements \( N \) and the number of bytes of the element type: \( M = N \times \text{sizeof(ElementType)} \).

Attention must be paid to how the data are copied and released. If an image is copied or a new one is generated, the memory space for both instances must of course be included in the calculation. Moreover, the memory requirement of the router itself and other project parts must be included in the calculation.

### NOTE

**Memory value with safety factor**

Always extend the calculated memory value with a safety factor. Also take into account the memory requirement of the other modules in the project.
So as not to use an unnecessarily large amount of space in the router memory, you should take care not to make any unnecessary deep copies and to release objects again that are no longer required (see Interface Pointer [131]). In order to find out how much memory is actually being used, the router memory can be monitored live as follows.

**Monitoring the router memory**

Even if the router memory is set in the project, this is a system-wide setting. Therefore, the router memory set in the project and currently used on the system can be viewed on the target system. To do this, right-click the TwinCAT icon in the Windows taskbar and select **Router > Info**.

![AMS Router Information](image)

**NOTE**

**Reduction of the available router memory**

The available router memory may be reduced through Exceptions or unreleased Interface Pointers. Therefore, the memory should be checked during the development, in particular in the case of an error. If not enough memory is available, restart the complete system and then check your program.

**5.1.2 Stack size**

The default size of 64 KB is usually sufficient for image processing applications. When using larger image formats or special functions such as the `F_VN_Clahe` [1081] this value is too small and the stack size must be increased. Then it is recommended to use 512 KB or larger directly.

It can be set in the TwinCAT system configuration under the Real-Time node in the **Settings** tab:
5.1.3 Network adapter

TwinCAT RT Ethernet adapters

Similar to the connection of EtherCAT devices, the TwinCAT real-time Ethernet driver must be installed on the corresponding network adapter to enable connection to GigE Vision devices. To open a corresponding installation dialog, click **TWINCAT > Show Realtime Ethernet Compatible Devices** in the Visual Studio menu.

Alternatively you can call the file `C:\TwinCAT\3.1\System\TcRteInstall.exe` on the target system; this also opens the corresponding installation assistant.
In the installation assistant, make sure that the corresponding network adapter is listed under **Installed and ready to use devices (realtime capable)**. If this is not the case, select an appropriate adapter from the list of **Compatible devices** and click **Install**.

### Specifying a persistent IP address

If the PC should not automatically specify the IP address of the network adapter, but instead you wish to manually specify a persistent IP address, you can do this in TwinCAT as follows:

1. Open the **IP-Stack** of the RT-Ethernet adapter in the I/O configuration.
2. Select the **Parameter (Init)** tab and display further parameters by clicking the **Plus (+)** in front of **TcIoIpSettings**.
3. Define the parameters **IpAddress** and **SubnetMask**.
4. Set the parameter **ManualSettings** to **TRUE**.
The same setting is automatically issued if you issue a manual IP address when creating a camera object. Alternatively, a persistent IP address can also be defined via the Windows Network Adapter properties.

**NOTE**

**Persistent IP addresses**

To prevent the IP address of the network adapter changing, define a persistent IP address.

**Operating multiple cameras on one port**

If you want to operate several cameras on the same RT Ethernet adapter, you must make sure that the UDP/IP settings allow this. Navigate to the UDP/IP stack under the RT Ethernet device and select the Parameters (Init) tab.

In the parameter list that appears, set $\text{IpMaxReceivers}$ and $\text{UdpMaxReceivers}$ such that they correspond to twice the number of cameras connected via this adapter. This is because there are two receivers per camera (GVCP and GVSP modules). The standard setting allows two camera connections, for which reason the parameters are set to $2 \times 2 = 4$. 
Advanced Settings

| Maximum Transmission Unit (MTU) | The maximum permitted size of the frame to be transmitted is 1514. Thus, jumbo frames are not currently supported. |

Other network protocols

We recommend limiting the use of the network adapter to the TwinCAT Ethernet protocol and Internet protocol version 4 (IPv4). Because if several protocols have access to the network adapter, they could send packets that could cause errors in the camera communication. The setting for this can be found in the Windows properties of the network adapter.

However, the IPv4 protocol is only required if DHCP is required to set up the camera connection. Otherwise you can deactivate this too and configure the IP address manually in TwinCAT in the IP-Stack.

NOTE

Deactivation of other network protocols

If possible, deactivate all network protocols on your Ethernet adapter except for the TwinCAT Ethernet Protocol. Otherwise, errors and delays may occur in the communication with GigE Vision cameras.

Diagnostics

In case of problems with the connection, especially with sent or received frames, you will find a tab with statistics information in the RT Ethernet adapter and in the IpStack the Parameter (Online) tab with individual counter information. Details can be found in the appendix/troubleshooting at Camera communication.

Promiscuous Mode

Promiscuous Mode is a setting in TwinCAT RT Ethernet adapters. It is required for debugging purposes with the Wireshark tool. This mode can cause problems when communicating with GigE Vision devices.

NOTE

Avoid Promiscuous mode

This mode can cause problems when communicating with GigE Vision devices. GigE Vision devices should therefore be operated without Promiscuous Mode!

You can check this setting by opening the corresponding RT-Ethernet adapter in Visual Studio and switching to the Adapter tab. The checkbox for Promiscuous Mode (use with Wireshark only) must be deactivated.
5.1.4 CPU cores and tasks

The configuration of tasks and CPU cores is not specific to TwinCAT Vision, but is the same for all TwinCAT components. For instructions on how to do this, see Real-time settings and Task settings. The issues and considerations relating to TwinCAT Vision are described below.

A TwinCAT Vision application requires tasks for various purposes:

- Cyclic task for executing the PLC
- Cyclic task for operating Vision devices
- Job Task for parallelized PLC execution (Optional; this type of task is described separately in the chapter Job tasks [61]).

Configuring PLC tasks

Basically, all cyclic tasks required for TwinCAT Vision are created automatically. When a PLC is created, a task with a cycle time of 10 ms is created. This means that image processing sequences in the PLC are repeated every 10 ms. You can change the task settings under SYSTEM > Tasks > [TaskName].
**Development environment**

<table>
<thead>
<tr>
<th><strong>Configuration of device tasks</strong></th>
</tr>
</thead>
</table>

The required tasks are generated automatically when creating Vision devices. The cycle times of these tasks can be adjusted as described above. For most cases it is recommended not to change the cycle times of these tasks. This is only necessary in special cases.

If the link of a Vision device with a task should have to be adjusted, this is done on the **Context** tab of the respective TcCOM objects of a **GigE Vision camera** or **File Source** object:

**WARNING**

**Floating point exceptions**

For PLC tasks that essentially execute Vision commands, it is urgently recommended that you deactivate the option "Floating point exceptions". Otherwise, unnecessary errors could occur with certain TwinCAT Vision API functions and lead to a system abort. However, this option must be active if Motion Control or Safety components are also executed on the same PLC task.

**NOTE**

**Floating point exceptions in Job Tasks**

If you also use **Job Tasks** for Vision applications and have disabled the "Floating point exceptions" in the PLC task, you must also disable this option in the assigned Job Tasks. Otherwise exceptions and system crashes can still occur.

**Watchdog stack**

This setting must be activated if **watchdogs** are to be used in the assigned PLC program.

**Cycle ticks**

The cycle ticks multiplied by the base time of the selected CPU core (by default 1 ms) produces the above-described cycle time of the PLC.

**Floating point exceptions**

This setting specifies whether or not TwinCAT checks for floating point exceptions.

For Vision applications it is recommended to deactivate this option. Besides that, these error messages do not indicate the exact reason for the exception.

**Watchdog stack**

**NOTE**

**Floating point exceptions in Job Tasks**

If you also use **Job Tasks** for Vision applications and have disabled the "Floating point exceptions" in the PLC task, you must also disable this option in the assigned Job Tasks. Otherwise exceptions and system crashes can still occur.

**Configuration of device tasks**

The required tasks are generated automatically when creating Vision devices. The cycle times of these tasks can be adjusted as described above. For most cases it is recommended not to change the cycle times of these tasks. This is only necessary in special cases.

If the link of a Vision device with a task should have to be adjusted, this is done on the **Context** tab of the respective TcCOM objects of a **GigE Vision camera** or **File Source** object:

The following TcCOM objects must be linked to a cyclic task, if they are used:
- Camera Image Acquisition [118]
- Camera Image Acquisition Simulation [118]
- ADS Communicator [118]
- File Image Acquisition [124]

If suitable tasks already exist under the selected device name when creating a Vision device, they can optionally be reused. In this case the following window appears; click **Yes** to confirm the reuse:

---

### Reuse of device tasks

When reusing device tasks, it should be ensured that they are not used by several Vision devices at the same time.

---

### Assignment to CPU cores

The assignment of tasks to CPU cores takes place in the TwinCAT project under **SYSTEM > Real-Time > Settings**. In multi-core systems, TwinCAT enables a distinction between Windows cores and isolated cores. With Windows cores, the operating system and the TwinCAT application share the processor time. The real-time portion can be limited to between 10% and 90%. In contrast, isolated cores are fully available to the TwinCAT application. For this reason, the use of isolated cores is recommended for Vision applications.

To display the current CPU core configuration of the target system, click **Read from Target**.

---

To change the CPU core configuration of the target system, click **Set on Target** and select the desired distribution of divided and isolated cores.
After that the target system must be restarted.

**Recommendation**
We recommend using at least one dedicated, isolated core for the Vision tasks (device and PLC). If there is a large load on the TcVnService due to the saving or loading of images (by camera streams, a File Source Control or direct from the PLC), we recommend reserving one core entirely for Windows or severely limiting the real-time portion of a core.

If multiple Tasks are assigned to a core, priority must be considered. As a rule, tasks with shorter cycle times should be assigned a higher priority.

---

### 5.1.5 Job tasks

Job Tasks are used by TwinCAT Vision to parallelize functions. Depending on the system in use, its configuration, the API functions in use, the image content and the number of job tasks, this can lead to a shortening of the execution time. The supported functions are marked in the API reference [131] by the comment “Can use available TwinCAT Job Tasks for executing parallel code regions.” The Vision Job Pool [125] is used to assign which Job Tasks are used by TwinCAT Vision. Job Tasks only need to be created and configured once; thereafter the parallelization takes place automatically in the runtime.

**Benefit of Job Tasks**
The actual benefit of Job Tasks must be checked in the individual case. This is dependent on the system in use, its configuration, the API functions in use, the image format, the image content and the number of Job Tasks.

**Use**

Like other tasks, Job Tasks can be created via the Solution Explorer > SYSTEM >, right-click Tasks > Add new item.... In the following Insert Task Dialog, select the type TwinCAT Job Task (Worker Task).

A corresponding core assignment can then be made via Solution Explorer > SYSTEM > Real-Time, like for all other tasks. Observe the notes under Restrictions and recommendations [62].
Floating point exceptions
This setting specifies whether or not TwinCAT checks for floating point exceptions. The same option should be selected here as in the Configuration of the corresponding PLC tasks [58]. Otherwise exceptions and system crashes can occur.

The assignment of which job tasks are used by TwinCAT Vision is made via the parameter list of the Vision Job Pool [125]. The Vision Job Pool can be found in the Solution Explorer below the Vision node, as soon as an application has been created there.

Prioritization

Prioritization of the Vision Job Pool access
If several programs that are executed in parallel share the Vision Job Pool, the program with the highest task priority is generally granted access to the Job Tasks. However, this assignment is dynamic, i.e. if the program with the highest priority is currently in a sequential processing section, lower-priority programs can use the Job Tasks. As soon as a higher-priority program can process tasks in parallel, the tasks of lower-priority programs are displaced. However, the master task of the lower-priority program is able to terminate a task started in parallel on its own, so that there is no need to wait for a new availability of Job Tasks.

Multiple tasks on one core
As indicated in the section on restrictions and recommendations, a Job Task should not share a core. Otherwise, the task priorities apply.

Restrictions

• Job Tasks can currently only be used for TwinCAT Vision.
• More than one Job Task may not be assigned to the same core.
• A Job Task should not be assigned to the same core as a Vision PLC task.
• A Job Task should not be assigned to the same core as the PlcAuxTask.

Recommendations

• Each Job Task should be assigned to a dedicated isolated core.
• However, if another cyclic task runs on the same core as a Job Task, the priority of the cyclic task should be higher than that of the Job Task. It should be noted that this restricts parallelization and, in the worst case, make it unavailable.
• No job task should be assigned to the default core.

If cyclic TwinCAT tasks or the Windows operating system claim computing time on the core at the same time, an analysis is hampered due to the varying load of the Job Task.

5.2 VISION node
Components that are relevant for image processing are configured via the VISION node and its sub-elements. It has tabs for Service Configuration [63] and Logging [64].

Displaying the VISION node
To display the VISION node in your TwinCAT project, right-click on your TwinCAT project and select Show Hidden Configurations > Show VISION Configuration. The next step is to create an Application node [65].
5.2.1 Service Configuration

The Vision Service offers a non-real-time counterpart for some real-time components (e.g. function blocks). For this purpose the Vision Service implements an ADS server, which enables access from both the real-time and development environments.

The Vision Service enables the TwinCAT Vision API function blocks to indirectly access the hard disk from the real-time environment, e.g. to load or save images.

The Vision Service is configured under VISION > Service Configuration.

**Vision Service status**

<table>
<thead>
<tr>
<th>Service on Target Machine</th>
<th>Target system on which the TwinCAT Vision Service is running.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection</td>
<td>Status of the connection between the development environment and Service. This is a good indicator of whether the Service is running properly.</td>
</tr>
<tr>
<td>Version</td>
<td>Version number of the TwinCAT Vision Service.</td>
</tr>
</tbody>
</table>

Before it is used for the first time, the Vision Service must be installed on the target system, see Setup instructions [16].

Manually starting the Vision Service

The Vision Service is normally started automatically together with TwinCAT. However, if the Vision Service is offline, it can be reactivated with a TwinCAT restart (click or ). If this does not work, or if the Service repeatedly fails to start, it should be re-registered with TwinCAT; see: Vision Service doesn't start [1471].
Default Directories

These paths define the point on the target system at which TwinCAT Vision, by default, should search for files and save them.

<table>
<thead>
<tr>
<th>Default Image Directory</th>
<th>This is the default path used by the function blocks to read and write images:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FB_VN_ReadImage [1304]</td>
</tr>
<tr>
<td></td>
<td>FB_VN_WriteImage [1310]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Default Container Directory</th>
<th>This is the default path used by the function blocks to read and write containers:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FB_VN_ReadContainer [1302]</td>
</tr>
<tr>
<td></td>
<td>FB_VN_WriteContainer [1308]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Default ML-Model Directory</th>
<th>This is the default path used by the function blocks to read and write ML models:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FB_VN_ReadMLModel [1306]</td>
</tr>
<tr>
<td></td>
<td>FB_VN_WriteMLModel [1312]</td>
</tr>
</tbody>
</table>

A complete list of all important paths can be found in the section Important paths [1467].

Vision Service Log

All important service events are written to the log file TcVnService.log on the target system, which can be found under C:\ProgramData\Beckhoff\TcVnService. In case of support questions regarding the TwinCAT Vision Service, the log file and the time at which the behavior occurred must be sent as well. The development system also features a logger [64] for all components under the Vision node.

Vision Service Cache settings

By default, the Vision Service caches the images it reads in order to provide the requested images more quickly the next time they are retrieved. This optimized mode of operation covers the requirements of most applications and is therefore recommended.

In case the image content changes but the file name remains the same, image caching can be disabled. To do this, open the Vision Service settings at C:\ProgramData\Beckhoff\TcVnService\settings.json on the target system, change the default value of the updateCacheOnReadSize parameter from 0 to 1 and save the file.

**NOTE**

**Inconsistent data**

Ensure that the same file is not accessed simultaneously by multiple processes, which can lead to file corruption or inconsistent data.

5.2.2 Logging

The Vision node has a logger which, depending on the Log Level, logs all important events of the TwinCAT Vision components underneath the VISION node. The settings can be found under VISION > Logging.
The following settings are available:

**Log Level**

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>Only error messages are logged.</td>
</tr>
<tr>
<td>Warning</td>
<td>Error messages and warnings are logged.</td>
</tr>
<tr>
<td>Information</td>
<td>Error messages, warnings and important events are logged. [default]</td>
</tr>
<tr>
<td>Verbose</td>
<td>Error messages, warnings and all important events are logged.</td>
</tr>
</tbody>
</table>

**Log file**

By default, the log file with the name TwinCAT.XAE.Vision.log is stored in the directory \%TwinCAT3DIR%\3.1\Components\Vision. This can be changed via the Logging tab. The same applies to the maximum size of a log file and the maximum number of files. If the limits are reached, old data is overwritten.

In case of support enquiries relating to behavior within the Vision node, please include the log file and the time at which the behavior occurred.

**Other loggers**

The Vision Service [63] also has a logger on the target system. In addition, there is a log file for the TwinCAT Vision installation. All relevant logging paths are listed in the section Important paths [1467].

### 5.3 Application node

A TwinCAT Vision configuration can contain several application nodes, all of which are located inside the VISION node. An application node is created as follows:

- Right-click on VISION Node → Add New Item...

- Enter a name for the Application node and click OK.
GigE Vision Camera nodes [66] and File Source nodes [119] are created under the application node. The purpose of application nodes is consolidation of thematically related Vision devices. The following figure shows an example structure with several Vision applications:

- VISION
  - Vision Job Pool
  - Station 1 - Detection
    - TopCamera
    - SideCamera
  - Station 2 - Quality Control
    - HighResolutionCamera
  - Station 3 - Code Identification
    - FutureCameraForCodeReading

In addition, application nodes enable the common export/import [129] of several Vision device configurations in a single XTI file.

5.4 GigE Vision Camera objects

Each GigE Vision Camera object represents a camera within a project. A project can contain several camera objects.

Creating a GigE Vision Camera object

Creating a camera is described in First Steps [25] under Creating a GigE Vision camera [36]. Refer here to the notes on the configuration of network adapters [54] and device tasks [58].

After creating a camera, the system automatically tries to read the GenAPI file from the camera. This file contains the complete description of the parameter tree of the camera (see GigE Vision [116]). On the basis of this description, the configuration tree is then made available in the Configuration Assistant [71] and the current parameter values are read by the camera.

Selecting a connected camera

To link a camera connected to the target system with a GigE Vision Camera object, the Camera Initialization Assistant can be opened by right-clicking the corresponding camera object and selecting Choose Target Camera. Alternatively, this dialog can also be opened via the Connection Options on the General [68] tab.
Camera Initialization Assistant

- **Camera Name**: Display of the name of the selected camera object.
- **Camera Ip Address**: IP address that is saved in the TcCOM parameters. The entry will be overwritten on selecting a camera and also updated in the module with OK.
- **Discovery Timeout**: The Discovery Timeout describes the time allowed for the response from the camera that was connected to the respective network port. Unit: milliseconds (ms)
  Default: 1000
- **Set Ip Address for Device**: Opens the following Force Ip Address dialog in order to set the IP address and subnet mask of a camera. The specified IP address and subnet mask are saved in the TcCOM parameter CameraIPAddress in the Image Acquisition Object.
- **Discover Devices**: The search procedure is started with Discover Devices.
- **Results table of the camera connections found**: List of the cameras found and display of the following camera information read out:
  model name, manufacturer, IP address, subnet mask, MAC address, version, serial number and user identification

---

**Development environment**

**TF7000 - TF7300**

*Version: 1.3*
Exporting/importing GigE Vision Camera objects

If you wish to save the configuration of a camera or import it into a project, follow the instructions in the chapter Exporting/importing configurations [129].

5.4.1 General

The General tab provides an overview of the camera object in TwinCAT and the connected camera.
**General object information**

The upper part contains general information about the camera object:

- **Name**: Camera1
- **Object Id**: 0x06020010
- **Type**: (empty)
- **Comment**: (empty)

Status of the camera instance

The middle part contains status information for the camera instance:

- **Connection**: Online
- **GenApi Descr.**: Available
- **Param. Values**: Up To Date
- **Simulation Mode**: Off
- **ADS Communication Module**: Off
- **Clear Messages**: (empty)

The following table explains the individual elements:
## Connection

Displays the connection status to the camera.

Via the button 🔄 the display can be updated with **Refresh Camera Status**. In addition, it is possible to open the dialog for selecting a camera with **Choose Target Camera**.

Possible status:

- **Unknown**: The connection to the camera cannot be checked. The camera module may not be in the operating state or creation of the camera module may have been incomplete.
- **Online**: The connection to the camera is established.
- **Offline**: There is no connection to the camera.
- **Disabled**: The camera node is disabled. There is no communication with the camera.
- **Not in Config Mode**: The connection status can only be displayed here in TwinCAT Config Mode.
- **Loading**: The camera module is initialized.

## GenApi Descr.

This indicates whether the GenApi structure (feature tree in the Configuration Assistant [71]) was successfully created. Status options:

- **Available**: GenApi structure was successfully created.
- **Not Available**: GenApi structure was not created. No features can be displayed.

The button 🔄 also allows the following actions:

- **Read GenApi Description from Camera**: The GenApi description of the camera is read. At the same time, all features in the configuration tree [75] are reset.
- **Export GenApi Description**: The GenApi description of the camera is saved locally.

## Param. Values

Status options:

- **Up To Date**: All features of the camera have values and are up-to-date (camera online).
- **Complete**: All features of the camera have values, but these may be outdated (camera offline).
- **Incomplete**: Not all features of the camera have values.

The button 🔄 also allows the following actions:

- **Write Local Changes**: All local parameter values that deviate from the camera values (marked in orange in the configuration tree [75]), are written to the camera.
- **Revert Local Changes**: Any local changes in the feature tree that have not yet been written to the camera will be reset.
Simulation Mode

In simulation mode, images from recorded camera streams can be sent to the PLC instead of a real camera supplying images (see Record/Playback [97]).

If the simulation mode is activated, the Image Acquisition Object is automatically deactivated and the Image Acquisition Simulation Object is activated.

If the simulation mode is deactivated, the Image Acquisition Object is automatically activated and the Image Acquisition Simulation Object is deactivated.

The corresponding object links (see Camera details [117]) will each be adjusted automatically.

Default: disabled.

ADS Communication Module

Required for the recording of a camera stream, see Record/Playback [97].

This checkbox merely activates or deactivates the Ads Communicator object of the Camera object.

*Note* Please note that if Ads Communicator object is activated, images captured by the camera will still be internally referenced for a short while so that they cannot be transformed into displayable images. For further details see Displayable images [142].

Default: disabled.

Clear Messages

Use this button to delete the messages written by this camera object.

The button is only active if messages are present.

Camera information

In the lower part, general camera information is displayed as far as it could be read from the camera (e.g. manufacturer and model name of the camera). This information can also be found in the configuration tree [75].

<table>
<thead>
<tr>
<th>Manufacturer:</th>
<th>Teledyne DALSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Name:</td>
<td>Nano-C2420</td>
</tr>
<tr>
<td>Device Version:</td>
<td>1.07</td>
</tr>
<tr>
<td>Ip Address:</td>
<td>169.254.7.4</td>
</tr>
<tr>
<td>Gateway:</td>
<td>0.0.0.0</td>
</tr>
<tr>
<td>Additional Info:</td>
<td></td>
</tr>
<tr>
<td>User Name:</td>
<td>S1193349</td>
</tr>
</tbody>
</table>

5.4.2 Configuration Assistant

Both the camera parameters and the communication behavior can be configured on the Configuration Assistant tab. The Configuration Assistant has four sections:
5.4.2.1 Image preview

The area on the left in the Configuration Assistant features a preview window and some operating elements.
In this image preview, only 8-bit integer images are correctly displayed.

**Lower bar**

Below the preview window are the controls for starting / stopping the preview and information about the cursor position on the image:

<table>
<thead>
<tr>
<th><strong>Start/Stop Acquisition</strong></th>
<th>Button for starting or stopping image acquisition.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Grab Single</strong></td>
<td>Button for triggering a single image.</td>
</tr>
<tr>
<td><strong>De-Bayering</strong></td>
<td>Images in Bayer format can alternatively be previewed as RGB images. <strong>Note</strong> If this option is activated, the next image from the camera will be converted, not the present one.</td>
</tr>
<tr>
<td><strong>Pixel Value</strong></td>
<td>The color or intensity value of the pixel at the mouse cursor position on the image is indicated here.</td>
</tr>
<tr>
<td><strong>Cursor Position</strong></td>
<td>Shows the mouse cursor position on the image. Please note that the zero point for images is always at the top left.</td>
</tr>
</tbody>
</table>

**Image acquisition and trigger settings**

Make sure that the image acquisition and trigger settings of the camera are selected in such a way that the camera reacts to the commands.

If, for example, the Trigger mode of the camera is active, the camera will not automatically send images at the start of the image acquisition. See the example image acquisition and trigger [1459] for further details.

**Upper bar**

Above the preview image there are some controls for facilitating the camera configuration and adjustment of the image display.

**Camera configuration**
### Set ROI

After clicking this button, an ROI can be drawn directly in the preview image by holding the left mouse button down. This ROI is written to the camera parameters, so that the next image is sent with the set ROI.

### Reset ROI

After clicking this button a previously set ROI will be deleted, i.e. image size and offset will be set to the default values of the camera.

### Whitebalance

Clicking will perform a white balance on color cameras. The camera must support the following GenAPI parameters for this: `BalanceRatioSelector` and `BalanceRatio` (or `BalanceRatioAbs` for older cameras). In addition, the camera image must contain a white (not overexposed or reflective) object, such as a sheet of paper, for white balance.

Following a white balance, the necessary values are saved in the named GenAPI parameter and can be adopted into the `Initialization Commands` or `UserSets`. Automatically persistent storage on the camera may be specific to the camera concerned, but is not a standard feature.

This button is only active if there is a valid color format (3-channel images as RGB or 1-channel images in the Bayer format).

### Image size

- **Fit to window**

  Adjustment options for the image display size. The image can be scaled between 20% and 6400% of the original size. Click **Fit to window** to automatically set the display size such that the preview makes the best use of the available space on the screen.

### Interpolation

Here you can choose between different interpolation methods to scale the image from the original size to the display size.

- **High Quality**
- **Nearest**

  - **High Quality**: Interpolation with high quality.
  - **Nearest**: Interpolation through adoption of the value of the nearest pixel.

### Right-click the image

- **Start Acquisition**
- **Stop Acquisition**
- **Save Current Image**

  - **Start Acquisition**: Start image acquisition.
  - **Stop Acquisition**: Stop image acquisition.
  - **Save Current Image**: Save image as bitmap, JPEG or PNG.
### 5.4.2.2 Configuration

The configuration tree contains three main groups:

- **TcCOM Parameters**: Parameterization of the communication behavior between TwinCAT and the connected camera. An explanation of these parameters can be found in the chapter on TcCOM parameters [94].

- **GigE Vision**: Information about the camera according to the GigE Vision standard, and IP configuration option. Depending on the camera, the settings may also be available in the GenAPI tree.

- **GenApi**: Camera parameterization based on the GenICam standard. The description of all camera features is read automatically from the GenAPI file by the camera. A feature can be an executable command, a (changeable) parameter or an organizational element. For some settings (e.g. image acquisition/trigger, binning, exposure time, etc.) see Camera configuration samples [1455]. Details of the parameters of the respective camera can be taken from the corresponding manufacturer's documentation. In addition, the GenICam standard [117] describes how the GenAPI feature tree is structured.

If you click a feature in the configuration tree, the properties of the feature will be displayed underneath the tree.

### Setting of parameters

Changeable features can be changed via a text box, a slider, a combo box or a checkbox. Read-only features are grayed out. The following buttons are available for every changeable configuration element:

- **Write Value**
  Writes the local change to the camera.

- **Reset Value**
  Deletes the local change if there is a difference between the local value and the camera value.
Load Value
Loads the current value of the parameter from the camera.

If the set parameter value deviates from the actual value on the camera, the parameter and higher-level elements in the configuration tree will be marked in orange. In addition, the deviation will be displayed in the properties window and attention drawn to it by a notice next to the camera status.

Offline changes
The parameters in the configuration tree are also changeable if the camera is offline or if the target is in Run mode. In this case the parameters are not written directly to the camera, but only saved in the project. As a result of this, it is possible to make offline parameter changes that can be synchronized with the camera at a later time.

Feature properties
The properties of the individual features are displayed underneath the configuration tree. This includes the size and address of the respective camera register, access rights and a description.
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the parameter.</td>
</tr>
<tr>
<td>Value</td>
<td>Local value of the parameter. If the local value deviates from the camera value, the camera value is additionally displayed in orange.</td>
</tr>
<tr>
<td>Address</td>
<td>Hexadecimal address of the parameter on the camera</td>
</tr>
<tr>
<td>Length</td>
<td>Size of the parameter in bytes</td>
</tr>
<tr>
<td>Endianness</td>
<td>Byte sequence of the parameter. This property must be observed when using the function blocks <code>FB_VN_ReadRegister</code> and <code>FB_VN_WriteRegister</code>.</td>
</tr>
</tbody>
</table>
| Sign       | • **Signed**: Parameter is signed  
• **Unsigned**: Parameter is unsigned                                                                                                                              |
| Cachable   | Specifies whether the parameter value is cached when no camera is connected.  
• **WriteThrough**: The value is written both to the camera and to the project cache and project memory.  
• **NoCache**: According to the GenAPI description of the camera, the value of this camera parameter should not be saved locally. Therefore, this parameter may be empty if there is no camera connection. |
| Access     | Access rights for reading and writing this parameter on the camera.                                                                                                                                          |
| RegisterValue | Register value of the parameter. This can deviate from the displayed value if it is a known type. Example:  
Name: PixelFormat  
Value: BayerRG8  
RegisterValue: 13                                                                                             |
| RegisterType | Data type of the register. By this type it is possible to read which of the read/write function blocks is to be used.                                                                                  |
| Visibility | Visibility level of the parameter. Possible values: **Beginner, Expert, Guru** and **All**  
The visibility of parameters can be set with the filter elements [78].                                                                                                         |
| DisplayName | Name of the feature to be displayed.                                                                                                                                                                        |
| Description | Purpose of the parameter                                                                                                                                                                                   |

These properties are required, for example, if a parameter is to be read or changed from the PLC. Essentially, attention must be paid to Address, Length, Endianness and RegisterValue. The corresponding function blocks can be found in the API reference under Camera Register Access [1282].

Depending on the data type of a register, further properties such as Min/Max/Increment may be added.

**Selectors**

Selectors are parameters that are used for the grouping and selection of several sub-features. In order to use a sub-feature, the Selector parameter must first be set to suit (both in the configuration assistant and in the PLC).

If a Selector parameter has no address property, it does not point to a camera register. It then serves merely to provide a clearer display in the configuration assistant. In this case the sub-parameters can be changed directly from the PLC.

In order to take into account all sub-parameters of a Selector when creating the Initialization Commands, iteration takes place automatically over all sub-features. A corresponding setting can be found in Actions & settings [78]. An example of a Selector is the Trigger Selector:
Filter elements

With the help of filter elements, features can be found quickly and easily in the configuration tree and in the Initialization Commands list:

1. Click on the + symbol to expand all tree elements. All parameters are visible.
2. Click on the -symbol to collapse all tree elements. Only the main groups are visible.
3. The search line filters the parameters by specific character strings or (hexadecimal) addresses.
4. In accordance with the GenICam standard, the four visibility levels Beginner, Expert, Guru and All can be selected. The visibility level of each feature is stored in the GenApi file of the camera.

![Filter Elements](image)

Fig. 1: *Note* The visibility level filter is only active on the Configuration tab and is disabled on the Initialization Command tab.

5.4.2.3 Actions & settings

Actions

At the top right is the action button , which can be used for further actions or additional settings:
## Development environment

### Read Camera Parameters

All parameter values are read from the camera and written to the configuration tree. This must first be confirmed by the following dialog:

![Confirmation dialog](image)

### Write Local Changes

All local parameter values that deviate from the camera values (marked in orange in the configuration tree), are written to the camera.

### Revert Local Changes

Any local changes in the feature tree that have not yet been written to the camera will be reset.

### Export GenAPI Description...

The camera's GenAPI description is stored locally in a path to be specified.

### Preferences...

The settings window described below is opened.

---

### Loading of saved GenAPI file not possible

The loading of a saved GenAPI file is not possible because a connection to the camera is needed in order to read some camera features.

### Settings

---

Configuration Assistant Preferences

![Settings window](image)

- **General**
  - Reset hidden messages
  - Overwrite Camera Parameters when processing new connection
  - Automatically calculate the MaxPacketNumber
  - Autostart Preview
  - 3000 ms ADS Communication Timeout

- **GigE Vision**
  - Use GVSP MinPacketDelay for streaming in config mode
  - Iterate Selectors automatically
**Reset hidden messages**

In some dialogs it is possible to remember the selected option with **Remember my decision**, e.g. in the query dialog of **Initialization Commands**.[81]

*If Remember my decision* was selected, the corresponding dialog will not appear the next time and the selected option will be executed automatically.

In this case, the setting can be undone using the **Reset hidden messages** button.

**Overwrite Camera Parameters when loading Project**

This setting controls whether saved camera values should be written directly to the camera when the project is loaded. This ensures that the camera is automatically in the state in which the project was last saved when it was configured.

This only works when the camera objects are initialized for the first time after the project has been loaded, provided that the camera is connected at the time and TwinCAT is in Config mode.

Default: activated

**Automatically calculate the MaxPacketNumber**

If this option is active, the maximum number of required (and allowed) packets for an image is automatically determined after reading out all camera parameters. The calculation is based on image size and pixel format. The calculated value is only written to the responsible GVSP object if it is greater than the current value. This functionality automatically adjusts the reserved space for an image to be received to the actual amount of image data.

Default: activated.

**Autostart Preview**

If this option is active, the live image of the camera is automatically displayed when the **Configuration Assistant** [71] is opened, if this is possible.

Default: disabled.

**ADS Communication Timeout**

The Communication Timeout relates to communication within the camera instance in TwinCAT. It only has to be increased if other timeouts in the GVCP or GVSP module also have to be increased.

Unit: milliseconds (ms)

Default: 3,000 (adapted to the camera)

**Use GVSP MinPacketDelay for streaming in config mode**

If this option is active, the MinPacketDelay from the GVSP module is used to reduce the bandwidth in Configuration Mode. If you switch this option off, the frame rate of the camera will be reduced instead.

Default: activated.
Iterate Selectors automatically

When this option is active, camera enumeration parameters that act as selectors are automatically iterated.

The value of a selector parameter is used as an index for addressing other parameters. By changing a selector value, the dependent parameters can be changed accordingly, thus achieving consistent parameterization of the camera.

Without this iteration, only the parameters of the currently selected selector would be described.

This applies to the settings in the configuration tree [75] as well as when creating the Initialization Commands [81].

Default: activated.

NOTE

Effect on Initialization Commands

If the Iterate Selectors automatically option is changed, the Initialization Commands [81] must then be recreated. Re-create the Initialization Commands by clicking the Create button on the Initialization Commands tab.

5.4.2.4 Initialization Commands

The purpose of the Initialization Commands is to save a specific parameter configuration for a camera and, if required, to set the camera to this configuration. This functionality is necessary because the camera always reflects the delivery state after a restart. The initialization can take place automatically when starting TwinCAT or manually by means of a method call in the PLC.

The initialization commands are comparable with the startup list of EtherCAT devices or with UserSets. The UserSet is a GenAPI feature that allows saving and automatically loading a configuration in the camera. UserSets are not required by the GigE Vision standard and are therefore not natively provided in all cameras.

In order to be able to provide such functionality uniformly for all GigE Vision cameras, TwinCAT Vision offers the possibility with the initialization commands. Of course, UsetSets can also be used as an alternative if available.

In addition to this general information, the initialization commands can be divided thematically into the following areas:

- Configuration [83]
- Creation [88]
- Troubleshooting [90]

Camera state after a boot process

After a GigE Vision camera has been booted, it contains by default the factory settings in the configuration. In order to restore a previous configuration status, the respective parameters must initially be written to the camera via UserSets, via Initialization Commands or in the background by the camera assistant.

The role of the Initialization Commands can be described as follows:
The read and write operations are triggered as follows:

<table>
<thead>
<tr>
<th>Action</th>
<th>Trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reading parameters from the camera</td>
<td>• Initial reading after setting up the camera</td>
</tr>
<tr>
<td></td>
<td>• Reading when restarting the TcCOM objects (e.g. when clicking Load Devices)</td>
</tr>
<tr>
<td></td>
<td>• Click on Read Camera Parameters</td>
</tr>
<tr>
<td>Writing individual parameters to the camera</td>
<td>• Individual changing and writing of the parameters by clicking Write Value in the configuration tree.</td>
</tr>
<tr>
<td></td>
<td>• Click on Write Local Changes</td>
</tr>
<tr>
<td></td>
<td>• When connecting a camera for the first time after loading a TwinCAT project, if Overwrite Camera Parameters when loading Project is activated in the Settings window.</td>
</tr>
<tr>
<td>Creating Initialization Commands in the camera object from current parameter values</td>
<td>• Click in the Initialization Commands tab on Create and Test the selected Initialization Commands</td>
</tr>
<tr>
<td></td>
<td>• Confirmation of the dialog when activating the configuration</td>
</tr>
<tr>
<td></td>
<td>If Remember my decision is selected, the Initialization Commands will always be automatically written or not written in future. This only applies to the respective camera and can be reset in the Settings window of the camera object by clicking Reset hidden messages.</td>
</tr>
</tbody>
</table>

**Note** The parameters for the test will be written to the camera at the same time.

| Writing parameters from the initialization commands to the camera | • If the Image Acquisition TcCOM object is switched to the OP state (start of TwinCAT), dependent on the parameter InitializationAutoMode in the Image Acquisition object. |
|                                                                 | • Call via a method of function block FB_VN_GevCameraControl. |
5.4.2.4.1 Configuration

The Initialization Commands tab consists of 4 areas, which are described individually below:

- Operating elements
- Configuration list
- Properties window
- Filter elements
Operating elements

The left group of buttons provides the basic functionality and is always active. The functions from left to right are:

<table>
<thead>
<tr>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create and test the selected Initialization Commands</td>
<td>The Initialization Commands are created with the current values of the configuration list. The parameters are also written to the camera and read back for testing.</td>
</tr>
<tr>
<td>Delete changes and Initialization Commands</td>
<td>The Initialization Commands created and stored in the TcCOM object are deleted. This also applies to all other changes that have been made to the list. This resets the entire list to its initial state.</td>
</tr>
<tr>
<td>Select all features of the CameraRegister category</td>
<td>All entries below the Camera Register category are selected.</td>
</tr>
<tr>
<td>Deselect all features of the CameraRegister category</td>
<td>All entries below the Camera Register category are deselected.</td>
</tr>
</tbody>
</table>

The right group of buttons provides special functionality that is only active for the entries under the Camera Register node. The functions from left to right are:

<table>
<thead>
<tr>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move element up</td>
<td>The selected element is moved up one list position.</td>
</tr>
<tr>
<td>Move element down</td>
<td>The selected element is moved down one list position.</td>
</tr>
<tr>
<td>Copy element</td>
<td>A copy of the selected element is created. The set value is also copied. The value of the copied entry cannot be changed later.</td>
</tr>
<tr>
<td>Remove copied element</td>
<td>Removes a previously copied element.</td>
</tr>
<tr>
<td>Rename copied element</td>
<td>Opens a window to rename the copied item.</td>
</tr>
</tbody>
</table>

Configuration list

The configuration list contains three main groups:

- **ForceIP**: If this option is active, a ForcedIp is performed when initializing the camera in the runtime. The CameraIPAddress from the Image Acquisition Module is used. Since the command is sent to the respective Mac address of the camera, the Initialization Commands must be regenerated with the changed address when the camera is replaced. The values used are only displayed here. Default: disabled.

- **UserSet**: Some cameras have a UserSet function that can be used to load an image of parameters stored in the camera in order to initialize the camera. If the camera provides this function, the desired UserSet can be selected here. If the option is active, the commands for selecting and loading the corresponding UserSet are written to the Initialization Commands. Default: disabled.

- **CameraRegister**: This group lists all registers that are declared as writable in the read GenApi file. This applies independently of the stored visibility, therefore also invisible registers are listed, which are displayed in the configuration tree only with the visibility level All. The initial order corresponds to the order in the GenApi file. If this option is active, all registers selected below it are used for the creation of the Initialization Commands. It is possible to select and deselect the individual registers and changes are done with the corresponding operating elements. Changes to the created Initialization Commands are identified in color and graphically. Default: activated.

When selecting or deselecting one of the groups or changing a setting below the groups, the Initialization Commands must be recreated. When using UserSets, the Initialization Commands contain the necessary UserSet commands instead of the parameters. The UserSet and CameraRegister options are mutually exclusive.
Selector Feature

If Iterate Selectors automatically is active in the settings [78], there is a separate group for each value of a Selector feature. Within this group, all registers that are dependent on the selector value and are writable are displayed. Therefore, the group and register names can occur more than once, in which case the values or register addresses differ. If the setting is not active, there is only one entry per selector.
Properties window

The Properties window displays the following information about the selected feature:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the parameter.</td>
</tr>
<tr>
<td>Value</td>
<td>Local value of the parameter. If the local value differs from the Initialization Commands value created, the feature is displayed in orange.</td>
</tr>
<tr>
<td>Address</td>
<td>Hexadecimal address of the parameter on the camera.</td>
</tr>
<tr>
<td>Length</td>
<td>Size of the parameter in bytes.</td>
</tr>
<tr>
<td>RegisterValue</td>
<td>Register value of the parameter.</td>
</tr>
<tr>
<td>DisplayName</td>
<td>Feature display name.</td>
</tr>
<tr>
<td>Description</td>
<td>Meaning of the parameter.</td>
</tr>
<tr>
<td>Information</td>
<td>Information whether the value is different (orange), the feature is added (green) or removed (black, strikethrough).</td>
</tr>
</tbody>
</table>

Filter elements

With the help of filter elements, features can be found quickly and easily in the configuration tree and in the Initialization Commands list:

1. Click on the symbol to expand all tree elements. All parameters are visible.
2. Click on the -symbol to collapse all tree elements. Only the main groups are visible.
3. The search line filters the parameters by specific character strings or (hexadecimal) addresses.
4. In accordance with the GenICam standard, the four visibility levels Beginner, Expert, Guru and All can be selected. The visibility level of each feature is stored in the GenApi file of the camera.
5.4.2.4.2 Creation

The creation of Initialization Commands can be initiated manually or automatically. In the automatic case a dialog appears if TwinCAT has detected in the background that the current parameter configuration deviates from the stored Initialization Commands.

The automatic check is performed when the configuration is activated or when the TwinCAT project is created, and only occurs if one of the three main groups "ForceIP", "UserSet" or "CameraRegister" is activated in Configuration [83].

If the dialog appears repeatedly without any previous change, this may be due to dynamic registers such as temperature values or counters whose access rights are declared as writable. Since the camera writes these values, they change over time or with each image capture, causing deviations. To remedy this, you can deselect these registers in the Initialization Commands configuration list so that they are no longer included in the change validation.

If the check of the Initialization Commands is to be performed while the TcCom modules required for this are not in the OP state, the window Reload Devices required is displayed. Otherwise, the window for creating the Initialization Commands appears directly.

If confirmed with Yes a Reload Devices is executed first to set all modules to OP state. Afterwards, the following dialog appears for each camera in the project, with the option to create the Initialization Commands. Selecting No will not perform any action and will skip the queries entirely.

The visibility level filter is only active on the Configuration tab and is disabled on the Initialization Command tab.
The manual creation of the Initialization Commands can be initiated via the button **Create and test the selected Initialization Commands** on the Initialization Commands tab.

### Full control?

If full control over the Initialization Commands is desired, deny the automatic query with activated **Remember my decision** and always create the initialization commands manually via the actions.

### Creation steps

The creation of the Initialization Commands comprises several actions that are performed in the background:

- Test writing of the Initialization Commands to the connected camera.
- Test reading of the previously written Initialization Commands from the connected camera.
- If value differences or errors are detected, a list with detailed information appears. If no critical errors have occurred, the user can still accept the error-free Initialization Commands.
- Write the Initialization Commands array to the ImageAcquisition TcCOM object of the camera.

Test writing and subsequent reading enables you to validate whether the camera accepts the initialization commands. One of the following windows is displayed during the generation:

![Fig. 3: Creation of the Initialization Commands via an automatic dialog](image1)

![Fig. 4: Creation of the Initialization Commands by manual action.](image2)

The following status information is displayed:

<table>
<thead>
<tr>
<th>Number of Features</th>
<th>Number of parameters with which the Initialization Commands were created.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Iteration</strong></td>
<td>Number of previous attempts to write the Initialization Commands to the camera. In case of success, this is &quot;1&quot;. In case of failure this value increases, as an attempt is made again to write the Initialization Commands successfully to the camera without the erroneous parameters. Normally no more than two iterations should occur; if they do, however, the concatenation in the TcCOM parameters of the ImageAcquisition must be deactivated.</td>
</tr>
<tr>
<td><strong>Number of Errors</strong></td>
<td>Number of errors that have occurred. These are described in more detail in the following window.</td>
</tr>
</tbody>
</table>
If the writing of the Initialization Commands to the camera works very quickly and successfully, the window disappears again immediately and is not visible.

Messages

After the successful creation of the Initialization Commands, the camera - due to the test writing - has exactly the configuration that will be written to the camera during operation later on. If errors occur during the test writing of the Initialization Commands to the camera, these are displayed in the following window:

Fig. 5: Uncritical errors when creating Initialization Commands

As long as no FATAL errors have occurred, the Initialization Commands can be created and saved anyway by clicking Create anyway. In this case the erroneous parameters will be excluded from the Initialization Commands and accordingly not written to the camera later on.

In the case of a FATAL error the Initialization Commands cannot be saved. Therefore, you cannot click Create anyway. First of all the cause of the error must be rectified.

These error cases and their rectification are described below Troubleshooting.

Warnings when creating the Initialization Commands

Warnings can also occur in addition to errors. These are mostly just an indication that the camera has not accepted and corrected a value.

5.4.2.4.3 Troubleshooting

For the basic camera communication and parameterization as well as the creation of the Initialization Commands, three components are mainly relevant:

- the content of the GenApi device description file
- the camera behavior
- and the GevImageAcquisition object behavior.

GenApi device description file
The GigE Vision [116] specification includes the GenApi module of the GenICam specification, which describes the functions supported by a camera. This description takes the form of an XML device description file, which must be created in conformity with the specification. The GenApi description provided by a camera is checked upon receipt. If non-compliant digits are included, they are reported as warnings or errors.

- A warning is issued, for example, if undefined registers are located in the reserved bootstrap area. These registers are subsequently ignored, but this does not usually lead to restrictions in operation. In individual cases it has to be checked whether the reported registers are needed.
- An error is reported, for example, if the described access rights do not match the camera behavior. That is, if the camera returns an error when reading registers marked readable or writing registers marked writable.
- If the GenApi description contains completely non-compliant passages, in rare cases the camera may not be able to operate at all.

**Camera behavior**

The camera behavior can have an impact if, as described earlier, the behavior differs from the GenApi description and if registers contain interdependencies or influence each other.

- Registers can, for example, be locked by other registers, so that these must first be activated or deactivated.
- Register values can be influenced by other registers or formulas which, for example, restrict the maximum value.
- The camera can change register values when other registers are written. For example, the camera may automatically reset the ROI when the pixel format is written.

**Refer to the manufacturer's documentation**

In case of problems with the camera behavior, compare precisely with the manufacturer's documentation. It is possible, for example, that the writing of individual ReadWrite registers is wrongly rejected by the camera.

**GevImageAcquisition object behavior**

The GevImageAcquisition object is GigE Vision certified as part of the TC3 GigE Vision Connector product and therefore behaves in a standard-compliant manner. Concatenating commands, if supported by the camera, and iterating selectors is active by default in the settings. This means that all possible register combinations are read or written when communicating with the camera. This can occasionally lead to problems due to the amount of commands or the timing. To remedy this, you can disable **EnableConcatenatedCommands** in the Image Acquisition Module or the **Iterate Selectors automatically** in the settings.

**Messages**

The components described above and their behavior have different effects on the creation of the Initialization Commands or on the possible messages.

**Warnings**

Warnings are displayed if the values read back differ from the values previously written. The differences can be caused by different camera behavior or dependencies between registers. This does not mean that the camera does not work with the Initialization Commands. Therefore warnings have no influence on the creation with **Create anyway**. Afterwards you should test if the reported registers have the correct value and the camera behaves as desired.
An example of when warning messages may occur is when an ROI is set. Depending on the current values in the camera, the warning occurs because the offsets are written first according to the GenApi order. If an offset greater than 0 is to be written, the width or height must be correspondingly smaller than the maximum value. In this case, the camera then resets all ROI values, causing the difference. In this case, the order of the ROI registers in the configuration list can be changed so that the width and height are written before the two offsets. If necessary in special cases, a register can also be copied, shifted in the order and thus written several times.

**Error**

Errors are reported if individual register values could not be written. As described above, this can have various causes, such as incorrect or changed limit values, addresses, write permissions or formula calculations at the time of writing. The affected registers are omitted when creating with **Create anyway** and checked off in the configuration list. Subsequently, it should be tested whether the omitted registers have an influence on the camera functionality.
If, for example, as can be seen in the screenshot, it is only the model name, or the register has not been changed and has a suitable default value, the camera can be used without restrictions despite this error message.

**Fatal errors**

Fatal errors only occur if the camera does not respond within a certain time. Then the process is aborted and one of the following messages appears. In these cases, the corresponding timeout must be adjusted.

**Timeout**

**Force IP or GVCP timeout**

In the course of writing the Initialization Commands, the Force IP command is also sent if it is configured. If the respective camera takes a long time to set the IP with this command, the following timeout error can occur:

![Camera: Forcelp failed](image)

Sending the Forcelp command failed. If you do not need to force the ip address you can deactivate it in the preferences window within the config tab. The following exception occurred:

Force ip failed: Received timeout error while performing ADS request (indexGroup: 0x01010080, indexOffset: 0x00001241)

Fig. 6: Error during Force IP

**Persistent IP addresses**

If no Force IP is used, it is recommended to set a persistent IP address.

In this case the Force IP can be deactivated in the configuration list. Alternatively, or in the event of a GVCP timeout, the TransmissionTimeout and/or the MaxTimeouts can be increased in the TcCOM parameters from GVCP Module. Note that the ADS Communication Timeout in the Settings window should always equal at least the product of the two preceding parameters.
5.4.2.5 TcCOM parameters

The parameters of the three TcCOM modules Image Acquisition, GVCP and GVSP are described below. Adjusting these parameters can help certain cameras to establish a stable connection. The configuration tree (p. 75) should be used to change the parameters.
Initially, only the parameter `InitializationAutoMode` is relevant here. The setting of the remaining parameters can, however, be necessary in the case of connection or initialization problems.

**Reloading in the case of parameter changes**

For parameter changes to become effective, the corresponding objects must be reloaded.

---

**Image Acquisition module**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CameraIPAddress</td>
<td>IP address of the camera. This IP address is used with the Force IP command. If this IP address is changed, the Initialization Commands [81] must be re-created in order for the change to become effective. In addition, the corresponding option in the Settings [78] must be activated for the Force IP command.</td>
</tr>
<tr>
<td>InitializationTimeout</td>
<td>Maximum time for transferring the initialization parameters to the camera. Depending on the camera used, the transmission may take longer, in which case the parameter has to be adjusted accordingly. Unit: Milliseconds (ms) Standard: 2,000</td>
</tr>
<tr>
<td>InitializationAutoMode</td>
<td>Selection of the way in which the initialization of the camera is to be executed automatically by the ImageAcquisition TcCOM object. Communication with the camera is impossible if it has not been initialized. AUTOINIT_SO: The camera is initialized during the start-up phase of the TcCOM object. In this phase only a limited amount of time is available. If initialization parameters are used, the initialization normally cannot be carried out within this start-up phase on account of the potential timeout. In this case the behavior of AUTOINIT_AFTER_SO is applied, even if AUTOINIT_SO is selected. AUTOINIT_AFTER_SO: The TcCOM object enters the OP state without initializing the camera. The camera is only automatically initialized when the connection to the camera is opened from the PLC (e.g. with <code>FB_VN_SimpleCameraControl.StartAcquisition()</code> or with <code>FB_VN_GevCameraControl.OpenCamera()</code>). NO_AUTOINIT: The camera is only initialized if <code>GevCameraController.InitializeCamera()</code> is called. This option is conceived for the case that full control over the initialization is required in order to execute it at a certain time. Standard: AUTOINIT_AFTER_SO.</td>
</tr>
<tr>
<td>EnableConcatenatedCommands</td>
<td>Option to select whether the associated commands should be linked or sent individually when writing several commands to the camera, for example the initialization parameters. The linking of the commands makes the transmission quicker, but may lead to problems with some cameras. Standard: enabled.</td>
</tr>
</tbody>
</table>
## GVCP module

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LocalPort</strong></td>
<td>The local UDP port through which the GVCP module communicates. If the value is zero, an automatically generated port is used. Standard: 0</td>
</tr>
<tr>
<td><strong>MaxTimeouts</strong></td>
<td>The maximum number of transmission timeouts that may occur before a GVCP packet is discarded. After each timeout the GVCP packet is resent. If the maximum number of timeouts is reached, an error is output. Standard: 3 (adapted to the camera)</td>
</tr>
<tr>
<td><strong>TransmissionTimeout</strong></td>
<td>After sending a GVCP packet, a response is awaited. If the response takes longer than the TransmissionTimeout time, it is assumed that the packet has not arrived, in which case the packet is retransmitted. Unit: Milliseconds (ms) Standard: 600 (adapted to the camera)</td>
</tr>
<tr>
<td><strong>HeartbeatRate</strong></td>
<td>The control channel of the application sends a heartbeat signal to the camera to check if the connection is still valid. The HeartbeatRate describes the time after which the next signal is sent. Unit: Milliseconds (ms) Standard: 1,000 Note The HeartbeatRate should not be set arbitrarily high, since many cameras trigger a timeout after a few seconds without heartbeat.</td>
</tr>
<tr>
<td><strong>DiscoveryTimeout</strong></td>
<td>The DiscoveryTimeout describes the time taken to wait for a Discovery command to respond. The Discovery command is sent when searching for cameras in the network. Unit: Milliseconds (ms) Standard: 1,000</td>
</tr>
</tbody>
</table>
GVSP module

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LocalPort</td>
<td>The local UDP port through which the GVCP module communicates. If the value is zero, an automatically generated port is used. Standard: 0</td>
</tr>
<tr>
<td>MaxPacketSize</td>
<td>The MaxPacketSize describes the maximum packet size of a GVSP packet. The header of the Ethernet frame is not included in this size. Unit: Byte Standard: 1,500 Note: At present 1500 is the limit for this parameter. No jumbo frames are possible.</td>
</tr>
<tr>
<td>MinPacketDelay</td>
<td>If the MinPacketDelay is increased, the GVSP packets are artificially delayed. This allows the bandwidth to be reduced. Unit: Timestamp counter unit(s) Standard: 0</td>
</tr>
<tr>
<td>MaxResendRequests</td>
<td>The maximum number of resend requests for a single GVSP packet that was faulty or took too long. An image is transmitted in a GVSP block consisting of several packets. The image is invalid if a packet of a block could not be successfully transmitted. Standard: 1</td>
</tr>
<tr>
<td>MaxBlockNumber</td>
<td>Maximum number of GVSP blocks that can be processed simultaneously. If the value is zero, the number is unlimited. Standard: 5</td>
</tr>
<tr>
<td>MaxPacketNumber</td>
<td>The maximum number of GVSP packets in a block. Blocks containing more packets cannot be interpreted as a valid image. Standard: 25,000 Note: The number is calculated automatically if the option &quot;Automatically calculate the MaxPacketNumber&quot; is active in the Preferences Settings.</td>
</tr>
<tr>
<td>BlockTimeout</td>
<td>The BlockTimeout describes the time that may elapse between individual packets of a block. Unit: Microseconds (μs) Standard: 5,000</td>
</tr>
</tbody>
</table>

5.4.3 Record/Playback

Camera streams can be recorded and input to the PLC on the Record/Playback tab. This can be helpful for the development in order to record a camera stream on the desk and repeatedly play it back offline at a later time. The real behavior of the camera hardware is thus simulated with the original images.

Either the simulation mode is activated, in which case camera streams can be played back, or the simulation mode is deactivated, in which case camera streams can be recorded. It is not possible to do both simultaneously.

A camera stream consists of individual images that are saved together with the corresponding time markers in their own format (.tcs). Through the additional saving of the time markers, the time behavior of the camera on the machine can be simulated with application realism. The image data are saved in precisely the format in which they are sent by the camera; no further conversion takes place.
System requirements
Use fast SSD hard disks and at least 8 GB RAM for the recording and playback of camera streams. If slower storage media are used, reduce the image rate if necessary via the parameter Image Offset.

(De)activating TcCOM objects
The activation/deactivation of the camera TcCOM objects should not be done manually, as further links have to be adapted when changing from the simulation mode and from the Ads Communication in the background.

The three areas of the Record/Playback assistant (Record, Playback, Stream files) are explained below.

Recording a camera stream:
The recording of a camera stream is only possible with an active camera connection. Accordingly, the simulation mode of the camera instance [66] must be disabled and the standard acquisition selected. The Ads Communicator TcCOM module is also required.

The settings for Record Camera Stream are located in the Record Camera Stream area:
### Development environment

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Name</td>
<td>Text box for the file name of the camera stream. The path used to save the file is defined in Selection of file path and streams [101].</td>
</tr>
<tr>
<td><img src="start.png" alt="Start Recording" /></td>
<td>Button for starting a recording. Only active if recording is possible.</td>
</tr>
<tr>
<td><img src="stop.png" alt="Stop Recording" /></td>
<td>Button for stopping a recording. Only active during recording.</td>
</tr>
<tr>
<td>Current Frames</td>
<td>Number of images in the currently recorded camera stream.</td>
</tr>
<tr>
<td>Current Size</td>
<td>File size of the currently recorded camera stream.</td>
</tr>
<tr>
<td>Advanced Options</td>
<td>Option to show and hide advanced options.</td>
</tr>
<tr>
<td>Image Offset</td>
<td>Specifies the image recording sequence. Depending on the writing speed of the storage medium, the image size and the frames per second, it may be necessary to skip images. Default: 1 (i.e. an attempt is made to record every image).</td>
</tr>
<tr>
<td>Max Frames</td>
<td>Maximum permitted number of images in the camera stream to be recorded. The recording will be ended automatically as soon as this number of images is reached. Default: 0 (i.e. no limit).</td>
</tr>
<tr>
<td>Max Size</td>
<td>Maximum allowed file size of the camera stream to be recorded in MB. As soon as this file size is reached, the recording is automatically stopped. Default: 0 (i.e. no limit).</td>
</tr>
</tbody>
</table>

### Enough free memory space?

Before recording a stream, make sure that there is sufficient memory space on your target system or limit the stream size if necessary with Max Frames and Max Size.

The following conditions must be met for a stream to be recorded:

- Simulation mode is disabled.
- An Ads Communication object exists and is activated.
- TwinCAT is in Run mode.
- The camera must be in the ACQUIRING status.

If these conditions are not met, the following notes and suggested solutions may appear:

- **Simulation Mode is currently active**
  - Disable Simulation
  - Simulation mode means a state in which a stream can be played back but not recorded. The state can be changed via the Disable Simulation link in the note. This also requires that the configuration is reactivated.

- **Simulation module is still operating. Try “Activate Configuration”**
  - Simulation mode has been switched off, but the change is only applied after the configuration has been activated.

- **Not in Run Mode**
  - The recording of a camera stream is only possible in TwinCAT Run mode. This warning is displayed in all other TwinCAT states.

- **Ads Communication Module not found!**
  - Enable ADS Communicator Module

- **Ads Communication Module is disabled**
  - Enable ADS Communicator Module
• The Ads Communication module is required for recording camera streams. It can be activated via the link in the note. If no ADS Communicator object exists yet, it can be added via Add New Item… on the camera object and Try to fix on the General tab [68]. The change is accepted after activating the configuration.

**Activating an ADS Communicator module in advance**

If you plan to record a camera stream at a later time without first stopping TwinCAT (e.g. during the runtime of a machine), you must activate the ADS Communicator module in advance.

If the Ads Communicator object is activated, it is possible that images from the camera may still be referenced internally in the object for a short time and therefore cannot be directly transformed into displayable images. For further details see Displayable images [142].

**Playback of a camera stream:**

In order to play back a camera stream, the simulation mode of the camera must be activated; this is possible either via the General [68] tab of the camera object or in the message mentioned below.

If you write the configuration with Write to Module into the TcCom object and save it in the project, these values are already available after the activation of the project and the simulation starts directly with the Start Acquisition command from the PLC. The advantage of this is that you do not have to manually reset the settings and start the stream every time.

The settings for playing back a camera stream are located in the Playback Camera Stream area:
### Development environment

<table>
<thead>
<tr>
<th>Selected File</th>
<th>Text box for displaying the selected stream file. The selection is made from the list of available streams.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start Playback</td>
<td>Button for starting the playback. Only enabled if playback is possible. The playback always begins with image 1, irrespective of where the stream may have been stopped previously.</td>
</tr>
<tr>
<td>Stop Playback</td>
<td>Button for stopping the playback. Only enabled while a stream is played back.</td>
</tr>
<tr>
<td>Trigger Mode</td>
<td>Selection of the Trigger mode, in which the frames in the stream can be triggered individually.</td>
</tr>
<tr>
<td>Trigger Image</td>
<td>Button for triggering an image. Only active when Trigger mode is selected and while a stream is played back.</td>
</tr>
<tr>
<td>Loop</td>
<td>Option to select whether the stream is to be played once or in an infinite loop.</td>
</tr>
<tr>
<td>Write to Module</td>
<td>Writes the current settings to the TcCom module. Stored are: Playback Mode, Playback Speed, Playback Strategy, Loop and the path of the selected stream file.</td>
</tr>
<tr>
<td>Read from Module</td>
<td>Reads the stored values from the module and sets these settings.</td>
</tr>
<tr>
<td>Advanced Options</td>
<td>Show / hide advanced settings.</td>
</tr>
<tr>
<td>Playback Speed</td>
<td>Acceleration/deceleration factor for playback speed. (&gt;1 acceleration; &lt;1: deceleration) Default: 1.00 = speed matches the recording</td>
</tr>
</tbody>
</table>
| Recovering Strategy | Strategy in the event that playback at the specified speed is not possible:  
  - Slow down: Playback speed delayed (default setting)  
  - Skip images: Skipping of images  
  - Abort on lag: Abortion of the playback (not recommended) |

The following conditions must be met for a stream to be played back:

- Simulation mode is enabled.
- TwinCAT is in Run mode.

The following notes and warnings may appear above the text field with the selected file name:

- Simulation mode is currently not active
  - Enable Simulation

- Simulation module is not operational. Try "Activate Configuration".

- Simulation Mode has been enabled, but the change only becomes active after the configuration has been enabled.

- Not in Run Mode
  - Playback of a camera stream is only possible in TwinCAT Run mode. This warning is displayed in all other TwinCAT states.

### Selection of file path and streams

The file path and stream selection is shown below:
### Development environment

#### Camera Stream File Directory
Currently set file path. This path is used both for reading already recorded streams and for writing new streams. Subfolders are also searched for stream files.

**Note** The file path always refers to the target system and not to the development system.

**Note** Make sure that this file path is not affected by write filters or similar. See [Important paths](#).

#### Browse…
This button opens a selection window for browsing the destination folder.

**Note** This is only possible when the development system is also the target system. Otherwise the following message appears:

![Open Camera Stream Directory](image)

**Cannot browse camera stream directory on remote target.**

Please enter a valid directory path for target "CP-35173E".

**OK**

#### Open directory…
Button for opening the specified file path in the local File Explorer. This can also be done by double-clicking in the free area in the file list.

**Note** This is only possible when the development system is also the target system. Otherwise the following message appears:

![Open Camera Stream Directory](image)

**Cannot open camera stream directory on remote target.**

Please open a file explorer on target "CP-35173E" and navigate to "C:\Users\Public\Tokision\CameraStreams\test".

**OK**

#### Restore default
Button for setting the default file path.

#### Refresh
Button to update the stream list.

**Name**
List of all camera streams in the selected file path, which are also used for stream selection for playback.

**Frames**
Number of images of which the respective camera stream consists.

**Size**
File size of the respective camera stream in MB.

**Bytes**
The size specification in bytes.
5.4.4 Camera Calibration

Area scan cameras can be geometrically calibrated with the help of a calibration pattern on the Camera Calibration tab.

During geometric camera calibration of area scan cameras, the imaging process is reconstructed from the image. For this purpose, points in the image are required whose positions in the real world are known. Calibration patterns with defined points are available from which images are taken.

Supported calibration patterns:

- Chessboard pattern [111]
- Symmetrical circles pattern [112]
- Asymmetrical circles pattern [113]
- Individual circles pattern [114]
Image acquisition / selection

As in the Configuration Assistant [71] there is a preview image for the image acquisition above which there are display options and below which there are controls for the image acquisition:

Displaying the preview image

The image is always displayed just as it is received from the camera. An image in the Bayer format will therefore be displayed unencoded as a grayscale image. At this point the set encoding is not yet applied to the image. The encoding is relevant only for the calibration process, but not for the preview image.

Above the preview window:

- **Interpolation of the preview image:**
  Combo box for selecting whether a high-quality or nearest-neighbor interpolation of the preview image should be performed. The selection only refers to the preview. The calibration is based on the original image data.

- **Size of the preview image**
  If Fit to window is selected, the image is scaled to the size of the preview window. Alternatively, a display size can be specified via the slider.

Below the preview window:

- **Start Acquisition / Stop Acquisition**
  Button for starting or stopping image acquisition. Streaming mode of the camera must be active.

- **Grab Single**
  Button for triggering a single image.

- **Pixel Value**
  Shows the color value of the pixel at the mouse cursor position on the image.

- **Cursor Position**
  Shows the mouse cursor position on the image. Please note that the zero point for images is always at the top left.

If the calibration pattern is within the camera’s field of view, the corresponding image can be selected for calibration. The Controls are located below the preview image on the left, the collection of images for calibration on the right.
• **Add current image >>**
  *Add current image* adds the currently displayed camera image to the collection for calibration.

• **Start / Stop Adding**
  Instead of a single image, a series of images can be added to the collection. The image series is started via **Start Adding**. Subsequently, images are added at intervals of the time set for **Interval** until the number entered for **Count** is reached or the addition of images is terminated via **Stop Adding**.

• **Load Images...**
  As an alternative to image acquisition, saved images can be loaded via **Load Images**. Note here that these were recorded with the same camera system, the same camera settings, the same lens settings and the same calibration pattern.

• **Save All Images...**
  **Save All Images** can be used to save the complete image collection.

• **Delete Selected**
  **Delete Selected** deletes the selected images from the collection.

• **Delete All**
  **Delete All** deletes all images from the collection.

---

**Image format**

At this point only Mono8, RGB24 and Bayer8 images may be used.

---

**Settings and calculations**

### Calibration

- **Calibrate Intrinsics**
- **Calibrate Extrinsic**

**Encoding**

- None
- Advanced Options...

**Algorithm**

- Pixel Contours

**Pattern**

- Chessboard
- Circles Sym.
- Circles Asym.
- Start Indent

**Intrinsic Options**

- Fix Center Point
- Fix Aspect Ratio
- p1 = 0, p2 = 0
- k3 = 0
- k4 = 0
- k5 = 0, k6 = 0

**Distance X**

- 4.6

**Distance Y**

- 7

**Color Inverted**

- **XTS 68x56x5**
For measurement tasks we recommend using a monochrome camera. However, if it should be necessary to calibrate a color camera with Bayer pattern, the corresponding Bayer pattern must be specified by Encoding. This is automatically selected by default.

**Algorithm**

For circle pattern detection, you can choose between different edge localization functions. The subpixel function can be used to improve the result depending on the image quality. However, this also means that the entire process takes more time. See the chapter Measurement [1215] for more information on the subpixel functionality.

- **Pixel Contours**
  Contour-based
- **Subpixel Interpolation**
  With Subpixel by interpolation
- **Subpixel Erf Approximation**
  With Subpixel after Erf Approximation

**Definition of the calibration pattern**

The calibration pattern used is selected and defined under Pattern:

- **Chessboard**
  A chessboard pattern is used
- **Circles Sym.**
  A symmetrical circles pattern is used
- **Circles Asym.**
  An asymmetrical circles pattern is used
  - **Start Ident**
    Checkbox to specify indentation of the first row of the circle pattern. In the standard system without a check mark, a row that is not indented is expected.
- **Width**
  Number of characteristic points in horizontal direction
- **Height**
  Number of characteristic points in vertical direction
- **Distance X**
  Distance in the world coordinate system of the feature points in horizontal direction
- **Distance Y**
  Distance in the world coordinate system of the feature points in vertical direction
- **Dropdown menu for individual circle patterns**
  Selection of an individual circle pattern (entries are only displayed if corresponding files are found in the specified folder)
- **Color inverted**
  Checkbox for selecting color inversion of the image. By default (no check mark), black objects are expected on a white background.

---

**Symmetrical pattern**

In the case of wholly symmetric patterns, the assignment of the points found may deviate from those defined or the coordinate origin may lie in a different quadrant. This can also occur if the pattern from the selected image is in a different orientation to that defined.

Therefore, after completion of the calibration procedure, you should check whether the coordinate system corresponds to your specification.

**Notes**

For information on patterns and parameter settings, please refer to the supported calibration pattern types:

- Chessboard pattern [111]
- Symmetrical circles pattern [112]
- Asymmetrical circles pattern [113]
- Individual circles pattern [114]
Intrinsic Options

- **Fix Center Point**: the principle point corresponds to the center of the image.
- **Fix Aspect Ratio**: the camera sensor is assumed to have square pixels, so that the aspect ratio of the pixels is the same in the x and y directions.
- **p1=0, p2=0**: it is assumed that no tangential distortions exist, and the corresponding coefficients are set to 0.
- **k3=0**: the radial distortion coefficient $k_3$ is assumed to be 0
- **k4=0**: the radial distortion coefficient $k_4$ is assumed to be 0
- **k5=0, k6=0**: the radial distortion coefficients $k_5$ and $k_6$ are assumed to be 0

Extrinsic Origin

Extrinsic Origin specifies the origin point of the world coordinate system in the calibration pattern. Depending on the selected calibration pattern, a corresponding background image is displayed.

- **Origin**: The selected origin refers to the corresponding point of the calibration pattern, not to the complete image.

Actions

- **Calibrate Intrinsics**: calculates the camera matrix and the distortion coefficients. All images of the selection are used for this purpose. These must all contain the defined calibration pattern in its entirety. If this is not the case, a negative **Reproj. Error** is displayed.

- **Calibrate Extrinsic**: calculates the rotation matrix and the translation vector. A prerequisite for this is the intrinsic calibration (camera matrix and distortion coefficients) and selection of the image where the points of the 2D calibration pattern are located in the subsequent measuring plane.

**Duration of a calibration**

The duration of a calibration increases with the number of reference points in the calibration pattern and the number of calibration images. Therefore, when choosing a calibration pattern, make sure that the duration remains within acceptable limits for you.
Deviation of the results

On account of the algorithm used, the results of the calibration may differ slightly from one another in the decimal place range from call to call. This concerns both the results of the calibration in the calibration assistant and those in the PLC as well as between both calibration variants.

Advanced Options

Circle pattern recognition can be reparameterized via the Advanced Calibration Options. This should only be done if there are difficulties with the default settings. For a better understanding of the setting parameters, see the parameterization of the function F_VN_DetectBlobs [1029].
Results

Reproj. Error

The Reprojection Error indicates the result of the intrinsic calibration:

• >0: valid result
  ◦ <1: good result
  ◦ >2: inaccurate result
• <0: No result and specific error number:
  ◦ -100: no image found
    The collection of images below the preview window is used for the intrinsic calibration. The live image of a camera is not taken into account.
  ◦ -200: the pattern could not be found in at least one image.
    All images in the collection below the preview window must show the complete pattern, including all feature points. For a better analysis, result images are created under %TwinCAT3DIR%\3.1\Components\Vision\_CalibrationAssistantOutput, which show the characteristic points that were found.
  ◦ -300: none of the existing images contain the selected pattern.
    See description for -200; in addition, we recommend checking the pattern configuration.
  ◦ -400: The user-defined description file for an individual calibration pattern is invalid.
  ◦ -500: An error has occurred.

Camera Matrix

The camera matrix describes the imaging process from 3D to 2D.

\[
\text{CameraMatrix} = \begin{bmatrix}
  f_x & 0 & c_x \\
  0 & f_y & c_y \\
  0 & 0 & 1
\end{bmatrix}
\]

• The parameters \( f_x \) and \( f_y \) indicate the scaling in x and y direction relative to the 3D object.
• The parameters \( c_x \) and \( c_y \) indicate the intersection of the optical axis in the image. Ideally, this should be in the center of the image. If Fix Center Point is selected under Intrinsic Options, the parameters \( c_x \) and \( c_y \) are automatically set to the center of the image.

Distortion Coefficients

Aberrations of the optical system can lead to distortions in the image. These can be partly compensated by the Distortion Coefficients during the coordinate transformation.
DistortionCoefficients = \([k_1, k_2, p_1, p_2, k_3, k_4, k_5, k_6]\)

- The coefficients \(k_1\) to \(k_6\) describe the radial distortions in the image.
- The coefficients \(p_1\) and \(p_2\) describe the tangential distortions in the image.

Rotation Matrix

The rotation matrix describes the orientation of the camera coordinate system relative to the world coordinate system and vice versa.

\[
\begin{bmatrix}
  r_{11} & r_{12} & r_{13} \\
  r_{21} & r_{22} & r_{23} \\
  r_{31} & r_{32} & r_{33}
\end{bmatrix}
\]

Translation Vector

The translation vector describes the displacement of the coordinate origin of the camera coordinate system relative to the world coordinate system and vice versa.

\[
\begin{bmatrix}
  t_x \\
  t_y \\
  t_z
\end{bmatrix}
\]

Write Results

The results are written to the Image Provider TcCOM object of the camera. This means that they can be called from the PLC via the methods of the function block F_VN_GevCameraControl [1324]. Even if a calibration has not yet been carried out, the description of the set calibration pattern will be written. This can be retrieved in the PLC with the method GetCalibPatternRef [1331].

Save To File...

The results can be saved in a txt file.

Undistort Selected

The Distortion Coefficients are applied to the selected image, and the result is displayed in a separate window.

Show Output

Opens the directory %TwinCAT3DIR%\Components\Vision\CalibrationAssistantOutput. Result images are created in this directory to facilitate an analysis of which characteristic points were found by the software and how they were assigned.

Alternatives in the PLC

If a camera has to be calibrated while the machine is running, the function F_VN_CalibrateCamera [944] is available for this in the PLC.

- Calibration in the PLC
  
  The duration of a calibration can be much longer than a PLC cycle. Therefore, make sure that no problems are caused in your machine sequence when you carry out a calibration in the PLC.

Calibration Assistant sample

Calibration Assistant [1448]
5.4.4.1 Chessboard pattern

A chessboard pattern consists of alternating light and dark squares, which together form a rectangular pattern. See chessboard example below with 4 x 8 intersections:

![Chessboard pattern example]

Requirements for the pattern

- The pattern should be applied to a surface that is as flat as possible (planar), e.g. a glass pane.
- The light and dark squares must
  - be arranged in an alternating pattern,
  - have the same size
  - and be square.
- A pattern must contain at least 16 fully visible squares, which together form at least 3 x 3 intersections.
- The background color around the pattern must form a 'quiet zone'.

Requirements for the image acquisition

- Exposure
  - Homogeneous illumination of the pattern
  - Good contrast between the light and dark squares in the image
  - No opening at intersections due to overexposure
- Figure
  - A square must be represented by at least 15x15 pixels.

Configuration in the Camera Calibration Assistant [103]

- **Width**
  
  Corresponds to the number of intersections in horizontal direction.

- **Height**
  
  Corresponds to the number of intersection points in vertical direction.

- **Distance x**
  
  Corresponds to the distance between two intersection points in horizontal direction.

- **Distance y**
  
  Corresponds to the distance between two intersection points in vertical direction.
### 5.4.4.2 Symmetrical circles pattern

A symmetrical circle pattern is composed of circles that are arranged evenly in rows and columns.

Other supported patterns with circles are asymmetric circle pattern [113] and individual circle pattern [114].

**Requirements for the pattern**

- The pattern should be applied to a surface that is as flat as possible (planar), e.g. a glass pane.
- The circles should be as large as possible.
- A sample must consist of at least 9 points.
- The circle centers must be applied at the same distance horizontally and vertically. The distance must be large enough to ensure that the circles do not touch each other and are recognized as separate objects.
- The background color around the pattern must form a 'quiet zone'.

**Requirements for the image acquisition**

- Exposure
  - Homogeneous illumination of the pattern.
  - Good contrast between circles and background.
- Figure
  - Perspective distortions should be kept to a minimum.
  - The standard specifications of the Advanced Calibration Options must be adhered to or adapted. By default, a circle must be represented by an area of at least 25 pixels and a maximum of 15000 pixels. The area should not be too small, otherwise interference from the background will be picked up.

**Configuration in the Camera Calibration Assistant [103]**

- **Width**
  - Corresponds to the number of circles per row.
- **Height**
  - Corresponds to the number of circles per column.
- **Distance x**
  - Corresponds to the distance between two circle centers in horizontal direction.
- **Distance y**
  - Corresponds to the distance between two circle centers in vertical direction.
• **Color inverted**
  By default, dark circles on a light background are assumed. If the pattern instead consists of light circles on a dark background, the colors can be inverted with **Color inverted** option.

5.4.4.3 **Asymmetrical circles pattern**

An asymmetric circle pattern is similar to a symmetric circle pattern [112], except that every second row is offset by half a column. This allows a smaller distance between two rows, allowing more lines in the same area.

**Pattern requirements**

- The pattern should be applied to a surface that is as flat as possible (planar), e.g. a glass pane.
- The circles should be as large as possible.
- A sample must consist of at least 9 points.
- The circle centers must be applied at the same distance horizontally and vertically. The distance must be large enough to ensure that the circles do not touch each other and are recognized as separate objects.
- The background color around the pattern must form a 'quiet zone'.

**Requirements for the image acquisition**

- Exposure
  - Homogeneous illumination of the pattern.
  - Good contrast between circles and background.
- Figure
  - Perspective distortions should be kept to a minimum.
  - The standard specifications of the **Advanced Calibration Options** must be adhered to or adapted. By default, a circle must be represented by an area of at least 25 pixels and a maximum of 15000 pixels. The area should not be too small, otherwise interference from the background will be picked up.

**Configuration in the Camera Calibration Assistant [103]**

- **Start indent**
  The first row is indented relative to the second.

- **Width**
  Corresponds to the number of circles per row.
• **Height**
  Corresponds to the number of circles per column.

• **Distance x**
  Corresponds to the distance to the nearest circle center in horizontal direction (across rows).

• **Distance y**
  Corresponds to the distance to the nearest circle center in vertical direction (within a column).

- **Color inverted**
  When a light pattern is used on a dark background.

### 5.4.4.4 Individual circles pattern

An individual circle pattern consists of circles, with the option of user-defined position and thus arrangement of the individual circle components. This also enables the use of complex custom 3D patterns for calibration.

**Requirements for the pattern**

- The pattern should be applied to surfaces that are as flat (planar) as possible.
- The circles should be as large as possible.
- A sample must consist of at least 9 points.
- The distance must be large enough to ensure that the circles do not touch each other and are recognized as separate objects.
- The background color around the pattern must form a 'quiet zone'.

**Requirements for the image acquisition**

- **Exposure**
  - Homogeneous illumination of the pattern.
  - Good contrast between circles and background.

- **Figure**
  - Perspective distortions should be kept to a minimum.
  - The standard specifications of the **Advanced Calibration Options** must be adhered to or adapted. By default, a circle must be represented by an area of at least 25 pixels and a maximum of 15000 pixels. The area should not be too small, otherwise interference from the background will be picked up.

**Requirements for the description file**

- The pattern must be described in the form of an XML file and stored on the development computer in the following folder:
  
  ```
  %TwinCAT3DIR%\CustomConfig\Vision\CalibrationPattern
  ```

- The XML file must be validated against the `TcVn CalibrationPattern.xsd` schema located in said folder.
- The description file must contain at least the DisplayName, the Origin and the 2D or 3D coordinates of the circle points.
Center point
As the center point is automatically given by the description of the points, the specification of the origin serves only as further information and is not evaluated in the calculation.

2D pattern
When specifying 2D patterns, a Z-coordinate of 0 is used internally for the calculation.

Constant offset
A constant offset can be implemented as a 3D pattern with constant Z-value.

Configuration in the Camera Calibration Assistant [ › 103]
If valid XML description files were found in the specified folder, they are displayed with the DisplayName that was read from the file.

- **Dropdown menu for individual circle patterns**
  Selecting an individual circle pattern.

- **Color inverted**
  Checkbox for selecting color inversion of the image. By default (no check mark), black objects are expected on a white background.

- **Extrinsic Origin**
  This is automatically set to the value stored in the file for the selected pattern and cannot be changed via the user interface.
  
  - The optional background image for the schematic representation of the calibration pattern is displayed accordingly, if the image name including file extension was specified in the description file and the image file is located in the same folder.

Sample template of a description file for individual circle patterns
The following code block can be used as a template for creating your own description files and contains samples of completed metadata and 2D point descriptions. The 3D point descriptions, which are also included, are commented out. If 3D is required, the Pattern2D block would have to be completely replaced by the Pattern3D block, i.e. mixing is not allowed.

Assignment of the points
It is helpful if the points are sorted by rows or columns and directly match the positions of the points in the calibration image. The points can be assigned more quickly and more reliably as a result.
<?xml version="1.0" encoding="UTF-8"?>
  <MetaData>
    <DisplayName>12345</DisplayName>
    <Origin>Center</Origin>
    <PreviewImage>PreviewImageSample.png</PreviewImage>
    <Description>This is a calibration pattern</Description>
    <Revision>V1.0</Revision>
    <Width>10</Width>
    <Height>10</Height>
    <Depth>4</Depth>
    <NumberOfPoints>9</NumberOfPoints>
    <PointDiameter>1</PointDiameter>
  </MetaData>
  <Pattern2D>
    <!-- 1st row -->
    <Point2D x="-5" y="-5"/>
    <Point2D x="-0" y="-5"/>
    <Point2D x="5" y="-5"/>
    <!-- 2nd row -->
    <Point2D x="-5" y="0"/>
    <Point2D x="0" y="0"/>
    <Point2D x="5" y="0"/>
    <!-- 3rd row -->
    <Point2D x="-5" y="5"/>
    <Point2D x="0" y="5"/>
    <Point2D x="5" y="5"/>
  </Pattern2D>
  <!-- <Pattern3D> -->
  <!-- 1st row -->
  <!-- <Point3D x="-5" y="-5" z="-2"/> -->
  <!-- <Point3D x="-0" y="-5" z="-2"/> -->
  <!-- <Point3D x="5" y="-5" z="-2"/> -->
  <!-- 2nd row -->
  <!-- <Point3D x="-5" y="0" z="0"/> -->
  <!-- <Point3D x="0" y="0" z="0"/> -->
  <!-- <Point3D x="5" y="0" z="0"/> -->
  <!-- 3rd row -->
  <!-- <Point3D x="-5" y="5" z="2"/> -->
  <!-- <Point3D x="0" y="5" z="2"/> -->
  <!-- <Point3D x="5" y="5" z="2"/> -->
  <!-- </Pattern3D> -->
</TcVnCalibrationPattern>

5.4.5 GigE Vision

TwinCAT Vision or the TC3 GigE Vision Connector product is certified to GigE Vision (Version 2.0.3). TwinCAT Vision can thus be used with all GigE Vision-compatible cameras. As the GigE Vision standard is subdivided with each version into features that are obligatory and optional, a certification does not mean that all features of the version are supported. As cameras for their part may also have additional features that do not meet the GigE Vision standard, e.g. data compression, no statement can be made about full support. It is therefore recommended to compare and test the required features individually.

The GigE Vision standard

GigE Vision is an interface standard for industrial image processing. It facilitates configuration and operation of industrial cameras. GigE stands for Gigabit Ethernet. By using the Gigabit Ethernet communication protocol, GigE Vision benefits from the following features:

- The default transmission rate is 1 Gigabit/s (extendable with 2.5 GigE, 5 GigE and 10 GigE).
- Cable lengths up to 100 meters without the need for amplification.
- Use of standard hardware and software interfaces for Gigabit Ethernet.

The GigE Vision standard defines communication with compatible devices via UDP/IP and consists of the following four elements:

- The GigE Vision Control Protocol (GVCP) defines how GigE Vision devices are to be addressed. It specifies the data channels and mechanisms for the transfer of images and configuration data between PC and camera.
- The GigE Vision Stream Protocol (GVSP) specifies the different data types and transmission methods used to transfer images from a camera to a PC. A Packet Resend option is also available to fix transmission errors.
- The GigE Device Discovery Mechanism makes it possible to find cameras in the network using search queries.
- An XML file contains the GenAPI description that defines all public functions of the camera. This description is based on the GenICam \[117\] standard.

The GenICam standard

Information on the GenICam standard and on the GenAPI description of cameras can be found on the EVMA website.

Compatibility

How to find out whether a camera is compatible with TwinCAT Vision: look to see whether the name "GigE Vision" is explicitly named in the product description or the GigE Vision logo is shown:

"GigE" versus "GigE Vision"

Note the difference between "GigE" and "GigE Vision"! Only GigE Vision compliant cameras are supported.

Connection to other devices is not supported

Moreover, the GigE Vision standard allows connection to other devices (such as strobe controllers). This function is not currently supported by TwinCAT Vision.

5.4.6 TcCOM objects

Each GigE Vision camera instance contains a number of TcCOM objects in the project tree.

Fig. 9: GigE Vision Camera TcCOM objects in the System Manager

TcCOM objects automatically managed by TwinCAT Vision

TcCOM objects are always managed automatically by TwinCAT Vision and do not have to be adapted or configured manually. Therefore the following information essentially serves to provide a more comprehensive technical understanding.
Development environment

**Cyclic tasks**
The TcCOM objects GevImageAcquisition, GevImageAcquisitionSimulation and GevAdsCommunicator must be linked with a cyclic task if they are activated. The corresponding configuration can be found on the Context tab of the objects. Normally, there is no need to manually link tasks, as tasks required when creating a Vision device are automatically created and linked. Details on creating and linking tasks can be found in the chapter on CPU cores and tasks.

**Automatic reloading**
If the TcCOM objects are to be automatically updated to the latest version during a version update of TwinCAT Vision, activate the corresponding checkbox on the Object tab:

- [ ] Auto Reload TMI/TMC

**GigE Vision Image Acquisition**
The GevImageAcquisition object coordinates the communication with a GigE Vision camera. It consolidates and manages two sub-objects that implement the two protocols of the GigE Vision standard:

- GigE Vision Control Protocol (GVCP)
- GigE Vision Streaming Protocol (GVSP)

Diagnostic information on the camera communication can be found on the Parameter (Online) tab of the two sub-objects.

It also uses a UDP/IP stack on an appropriate network adapter to communicate with a GigE Vision camera. Diagnostic information can also be retrieved in these objects. Details on both diagnostic options can be found in the Appendix/Troubleshooting at Camera communication.

Fig. 10: RT-Ethernet adapter and IP stack in the System Manager

Image acquisition and the two protocols (GVCP and GVSP) are parameterized in the Configuration Assistant in the configuration tree. A description of the parameters can be found in Chapter TcCOM parameters. This object is deactivated when camera simulation is active.

**GigE Vision Image Acquisition simulation**
The GevImageAcquisitionSimulation object simulates the streaming behavior of a GigE Vision camera and is deactivated by default. If the simulation mode is active, the GevImageAcquisition object is deactivated instead.

<table>
<thead>
<tr>
<th>ActivateBlockIDCorrection</th>
<th>This setting can be used to specify that artificially generated continuous block IDs should be used instead of original IDs. Default: disabled</th>
</tr>
</thead>
</table>

**GigE Vision Ads Communicator**
The GevAdsCommunicator object is required when recording a camera stream. It sends corresponding images from the router memory to the TwinCAT Vision service via ADS. In contrast to saving images via the function block FB_VN_WriteImage, a separate TcCOM object is required for the record functionality, since the recording runs independently of the PLC processing.

In Config mode, a further instance of the GevAdsCommunicator module is used so that all Camera Assistants can communicate with the camera.
PublishImages
With this setting, the Ads Communicator sends every nth image to the TwinCAT Vision service. This setting is automatically set when activating a stream recording and depends on the Image Offset parameter on the Record/Playback tab. It should not be set manually.
Default: 0

GigE Vision Image Provider
The GevImageProvider object represents the interface between a camera and corresponding function blocks in the PLC. These include FB_VN_GevCameraControl, FB_VN_SimpleCameraControl, FB_VN_ReadRegister and FB_VN_WriteRegister, and FB_VN_ReadMemory and FB_VN_WriteMemory. To this end, the object in the Symbol Initialization tab of the project instance must be linked with the desired PLC symbols of the function blocks. If several camera instances exist in the project, ensure that the correct GevImageProvider module is selected based on the name.

ImageQueueSize
Number of acquired images that are to be buffered if they are not accepted directly by the function block in the PLC.
Default: 1

Linking the TcCOM objects
The following overview graphic shows a simplified relationship of the different objects that communicate with one another or retrieve data.

Fig. 11: Architecture of the GigE Vision Camera functionality
The linking of the TcCOM objects differs depending on the setting of the simulation mode and the Ads Communicator object (see General tab). The corresponding link setting can be found in the Interface Pointer tab of the individual objects. By default, the link is automatically created based on the settings in the Camera Assistant and should not be changed manually. If the links were changed by mistake or the TcCOM objects were added to the camera afterwards, the links can be repaired by clicking Try to fix in the General tab.

Fig. 12: Note on the invalid linking of TcCOM objects

5.5 File Source objects
The File Source represents an alternative image source to the camera. It enables stored images to be loaded from the file system to the TwinCAT real-time instead of live images. Note that compared to the Record/Playback function of a GigE Vision Camera object, a File Source is not intended for the loading of continuous camera streams, but for the loading of individual images.
Images can be used as long as they are saved in a standard image format, e.g. bmp, jpg, png.
Compressed image formats

When using compressed image formats, it depends on whether the applied method is lossy, as the image data would then no longer correspond to the original image data of the camera. Therefore, we recommend using BMP as the storage format.

Grayscale images can be color images

Images that look like grayscale images can be saved with several channels. Pay attention to this when calling conversion functions.

The function blocks FB_VN_FileSourceControl [1316] and FB_VN_SimpleCameraControl [1348] exist so that images can be received in the PLC. Communication to the File Source Control takes place via these function blocks in a similar way to communication to a camera. An alternative to loading individual images into the TwinCAT real-time is in addition the function block FB_VN_ReadImage [1304].

Image delivery modes

The File Source Control has two modes for providing images:

1. Streaming mode: Like a streaming camera, new images are provided in a specified cycle.
2. Trigger mode: A new image is provided with a trigger signal. This can be triggered either by a software trigger or by manually actuating a button.

The settings and the image selection are done under File Source Control [120].

Creating a File Source node

Creating a File Source node is described in First Steps [25] under Creating a File Source Control [33].

Export/Import of a File Source node

If you wish to save the configuration of a File Source or import it into a project, follow the instructions in the chapter Exporting/importing configurations [129].

5.5.1 File Source Control

The configuration of the File Source Control takes place on the tab of the same name: VISION Node > File Source Node > File Source Control tab.
Manage image selection

The images to be used by the File Source Control are managed via a file list and the associated controls:

- **Read from Target**
  After creating a File Source, **Read from Target** must be pressed once and the configuration activated so that images can be added.
Development environment

<table>
<thead>
<tr>
<th>Checkbox</th>
<th>The checkbox in the first column indicates whether an image is used or not.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preview</td>
<td>The size of the preview image depends on the column width.</td>
</tr>
<tr>
<td>File</td>
<td>File name of the image, including data type.</td>
</tr>
<tr>
<td>Format</td>
<td>By default, images are sent in the saved format. Alternatively, it can be specified to convert this into a 1-channel grayscale image or 3-channel RGB image in advance and then send it. Note: We recommend that you explicitly specify the format for each image.</td>
</tr>
<tr>
<td>Read From Target</td>
<td>The information which images are stored on the target system for the associated File Source Control is read out.</td>
</tr>
<tr>
<td>Add Files</td>
<td>Images from the development system can be selected via a file explorer. These are then copied to the target system in the background, so that they are available there independently of the development system. Please note that only files with unique names can be used. In addition, only image formats, i.e. not video formats, can be added here. If you want to load a camera stream (*.tcs) into the PLC, please use the Record/Playback [97] function. Note: Unless configured otherwise, the image files are stored on the target system in the path &quot;C:\Users\Public\TcVision\FileSources&quot;. Make sure that the corresponding path is not affected by other software such as WriteFilter.</td>
</tr>
<tr>
<td>Add Directory</td>
<td>A directory of the development system from which all images are to be used can be selected via a file explorer.</td>
</tr>
<tr>
<td>Clear All</td>
<td>All images will be removed from the file list.</td>
</tr>
<tr>
<td>Move Up / Move Down</td>
<td>A selected image is moved one position up/down in the file list.</td>
</tr>
</tbody>
</table>

Context menu

Use CTRL+click to select several images in the file list. By right-clicking on one of the selected images, you can then adjust the settings for all selected images:

![Context menu for the image selection in the File Source Control](image)

Fig. 15: Context menu for the image selection in the File Source Control
Save As... | Opens a file explorer dialog for saving the selected images.
Remove | Deletes the selected images.
Set Format | Sets the image format for all selected images:
- Original Format from File: The format in which the image is stored is used.
- 24-bit RGB: The images are sent as 3-channel RGB images by the File Source.
- 8-bit Monochrome: The images are sent as 1-channel grayscale images by the File Source.
Sort by | Sorts all images by file list:
- Name (Ascending): ascending by name
- Name (Descending): descending by name
Select | Selects the highlighted images so that they can be sent via File Source Control.
Unselect | Deselects the highlighted images so that they are not sent via File Source Control.

Image transfer settings

Like a streaming camera, the File Source Control can provide a new image in a specified time cycle or after calling a software trigger.

Cycle time: 10ms  Trigger Mode

Fig. 16: Setting options in the File Source Control

| Cycle Time | If no Trigger mode is selected, the images are sent to the real-time context at the Cycle Time interval. The order of the images according to the file list is taken into account. This parameter is comparable to the frame rate of a camera.
Note: The maximum speed for the loading of images depends on the system and the images. The Cycle Time is merely the desired value.
| Trigger Mode | If Trigger Mode is selected, an image is sent after a corresponding software trigger is called. With the standard software trigger, the order of the images according to the file list is observed. Alternatively, a specific image can be triggered based on the file name.

Online

Fig. 17: Operation in the File Source Control

Start Sending | Start Sending in the PLC corresponds to the StartAcquisition[1320] method of the function block FB_VN_FileSourceControl[1316]. If no trigger is selected, the images are then sent via File Source Control at the Cycle Time interval. If the trigger is selected, File Source Control responds to a software trigger and sends the images according to a trigger signal.

Stop Sending | Stop Sending in the PLC corresponds to the StopAcquisition[1321] method of the function block FB_VN_FileSourceControl[1316] and terminates the sending of images via File Source Control.

Trigger Image | Trigger Image in the PLC corresponds to the TriggerImage[1322] method of the function block FB_VN_FileSourceControl[1316]. This can be used to trigger images manually. This button is only available if Trigger Mode is enabled and image transfer via File Source is active.
5.5.2 TcCOM objects

Each FileSource instance contains two TcCOM objects in the project tree.

- FileSource1
  - FileSource1 Image Acquisition (TcIoFileImageAcquisition)
  - FileSource1 Image Provider (TcInFileImageProvider)

Fig. 18: File Source TcCOM objects in the System Manager

TcCOM objects are automatically managed by TwinCAT Vision

TcCOM objects are always managed automatically by TwinCAT Vision and do not have to be adapted or configured manually. Therefore the following information essentially serves to provide a more comprehensive technical understanding.

File Image Acquisition

The TcIoFileImageAcquisition object coordinates the loading of images from the hard drive for FileSource instances. This object contains the following parameters (in addition to CycleTime and TriggerMode, which can be set in the File Source Control):

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BaseDir</td>
<td>File path on the target system under which images added to the selection list are stored. Standard: C:\Users\Public\TcVision\FileSources&lt;UniqueId&gt;</td>
</tr>
<tr>
<td>ServiceTimeout</td>
<td>Timeout for communication with the TwinCAT Vision Service in milliseconds. Standard: 5000</td>
</tr>
</tbody>
</table>

Cyclic tasks

The File Image Acquisition object must be linked with a cyclic task. The corresponding configuration can be found in the Context tab of the objects. Normally, there is no need to manually link tasks, as tasks required when creating a Vision device are automatically created and linked. Details on creating and linking tasks can be found in the chapter on CPU cores and tasks.

Automatic reloading

If the TcCOM objects are to be automatically updated to the latest version during a version update of TwinCAT Vision, activate the corresponding checkbox on the Object tab:

- Auto Reload TMI/TMC

File Image Provider

This object represents the interface between the FileSource object and the FileSource function block in the PLC. The ObjectID variable of the function block FB_VN_FileSourceControl or FB_VN_SimpleCameraControl must be linked with the corresponding File Image Provider TcCOM object on the Symbol Initialization tab of the project instance after the compilation of the PLC code. If several FileSource instances exist in the project, care must be taken on the basis of the name that the correct object is selected. This object contains the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageQueueSize</td>
<td>Number of loaded images that are to be buffered if they are not accepted directly by the function block in the PLC. Standard: 1</td>
</tr>
</tbody>
</table>

Linking the TcCOM objects

For a File Source device only the Image Provider has to be linked to Image Acquisition. By default, this link is created automatically and should not be changed manually. The following overview graphic shows a simplified relationship of the different objects that communicate with one another or retrieve data.
5.6 Vision Job Pool

The Vision Job Pool is used to manage Job Tasks [61] for image processing tasks. This enables the parallelized execution of selected TwinCAT Vision algorithms, which are marked in the API reference [131] by the following supplement: "Can use available TwinCAT Job Tasks for executing parallel code regions".

Clicking on the Vision Job Pool opens a window. The assignment of the Job Tasks takes place in this window on the Parameter (Init) tab. In the first row, the Unit column indicates how many Job Tasks are to be assigned. The array elements under Job Tasks can then be assigned a Job Task using the dropdown menu in the Value column. Each Job Task may only be assigned to one element.

![Diagram of Vision Job Pool]

**NOTE**
Be sure to refer to the restrictions and recommendations [62] when creating Job Tasks.

5.7 ADS Image Watch

With the ADS Image Watch, images of the type ITcVnDisplayableImage [383] can be retrieved from the PLC and displayed in the development environment. Open the ADS Image Watch via the Visual Studio menu TwinCAT > Windows > ADS Image Watch.
Image selection

The image selection is made using the selection fields from left to right.
<table>
<thead>
<tr>
<th><strong>Target system</strong></th>
<th>ADS devices are addressed via the ADS-AmsNetId and the ADS port. The ADS-AmsNetId unambiguously identifies the target system that must be selected first. The local system is selected by default. However, you can select any system for which a route has been created; see Choose Target System [27].</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port</strong></td>
<td>The individual devices on the system are identified by ADS port numbers. A fixed port number range is defined for the PLC runtimes, so that the selection list is pre-filtered based on these. If only one PLC port exists on the target system when this is selected in the ADS Image Watch, the PLC port is selected automatically.</td>
</tr>
<tr>
<td><strong>Image variable</strong></td>
<td>All variables of type [TcVnDisplayableImage [383] are displayed here and can be selected, regardless of whether the variable points to an image with a supported image format [128] at the time.</td>
</tr>
<tr>
<td><strong>Size</strong></td>
<td>Without additional settings, the image is scaled to the size of the current window (Fit To Window). However, it can optionally be set to a size relative to the original image. Please note that the image is scaled down by the PlcAux task in the real-time environment before transmission via ADS if the selected size is smaller than the original image. If the image is then saved, it does not have the original size.</td>
</tr>
<tr>
<td><strong>Stop / Play</strong></td>
<td>The alternating Stop / Play button can be used to stop or restart the updating of the image display. If the pause icon is visible, the update is active and images are continuously received. If the play icon is visible, the update has stopped and the display of the last image is maintained. No more images are received. This makes it possible to analyze the image and every detail, e.g. by zooming in.</td>
</tr>
</tbody>
</table>

**Options**

The currently displayed image can be saved using the Action icon at the top right next to the image size, then Save Current Image.

Furthermore, the background around the display image can be changed between light and dark with Invert Background. This can be helpful with very bright or dark images, so that the image content or image edge can be clearly seen and distinguished by a high contrast to the background.

**Image size**
The image is saved in the size that the ADS Image Watch queries from the PLC. In order to save the image in the original size, the image must be set to at least 100% or the window pulled to at least the original image size. If the image size is set to larger than 100%, the image will also be saved in the original size. With a smaller setting or with Fit to Window and too small a window, the image will be stored in the correspondingly reduced size.
Image formats supported

The following image formats can be displayed in the following color spaces in the ADS Image Watch:

- **1-channel images** are interpreted as grayscale images. All images with pixel types of enum ETcVnElementType [186] are supported.
- **3-channel images** are interpreted as RGB color images. Images with 8 bits or 16 bits per channel are supported; this corresponds to the pixel types [186] USINT, SINT, UINT and INT.
- **4-channel images** are interpreted as RGBA color images. Images with 8 bits per pixel per channel are supported here; this corresponds to the pixel types [186] USINT and SINT.

Other image formats

Other image formats and color spaces can be processed in the PLC as ITcVnImage. To display them, however, they must first be converted to one of the displayable formats by means of F_VN_ConvertColorSpace before they are converted into a ITcVnDisplayableImage. If this is not done, the images will either be displayed in the wrong color representation or not at all.

Displaying 16-bit images

16-bit images are transferred from the PLC to the ADS Image Watch as 8-bit images. Therefore the pixel values are only displayed in the status bar in increments of 256 (0, 256, 512, etc.), but nevertheless relate to 16 bits: (0: Minimum, 65,280: Maximum).

Displaying (L)REAL images

(L)REAL images are transferred from the PLC to the ADS Image Watch as 8-bit images. For this purpose the value range [-1, +1] is scaled to [-127, +127] and transmitted as an integer. Floating point values outside these limits will be interpreted as -1 or +1. In the ADS Image Watch the pixel values to be transmitted are divided by 127 again so that the floating point values are displayed in the range [-1, +1].

Status Bar

The status bar below the image shows the following information (from left to right):

![Status bar in the ADS Image Watch](image)

Fig. 20: Status bar in the ADS Image Watch

- Color values/intensity values at the mouse pointer position on the image or number of pixels in the corresponding container at the mouse pointer position in the histogram
- Image coordinates at the mouse pointer position on the image or container at the mouse pointer position in the histogram
- Image resolution

Histogram

When the **Show Histogram** checkbox is checked, the histogram of the image is displayed at the bottom left of the image. This can have several channels that are displayed superimposed, depending on the image type.
5.8 Exporting/Importing nodes

TwinCAT offers the option of saving individual configuration nodes such as I/O or MOTION elements in a separate XTI file or importing them from this file. This is also possible with application nodes as well as with GigE Vision Camera and File Source objects. In this way, configurations for Vision devices can be saved independently of the project and re-imported elsewhere.

Export

- Right-click on the node to be exported and select **Save <Name> As…**

- Select a location for the XTI file.

Import

- Depending on whether you want to import an entire Application node or just a single Vision device, the following steps should be performed on the VISION node or on a corresponding Application node.

- Right-click on the node into which you want to import the configuration and click **Add Existing Item…**
• Select the XTI file you want to import.
• If necessary, apply system-specific settings (e.g. network adapter) for the imported Vision devices.

**Duplication of camera configurations**

Please note that a successful transfer of the same configuration for several cameras can only be guaranteed if both the camera model and the firmware are identical. Otherwise problems may occur, e.g. if the register addresses of the camera are changed.

**Independent project files**

It is also possible to save the configuration nodes in a file of their own within the project, rather than exporting them completely. In this case the configuration for the corresponding nodes is not saved in the TwinCAT project file, but in a separate XTI file. This helps, for example, with version control systems, so that when a configuration is modified, the entire project file is not changed.
6 API reference

This chapter describes the TwinCAT Vision API, which provides options for programmatic use of the TwinCAT Vision library. As TwinCAT Vision brings along some innovations, the software concept is explicitly explained:

- Software concept [► 131]
- Data types [► 150]
  (aliases, enums, structures)
- Interfaces [► 223]
  (images, containers, etc.)
- Functions [► 401]
  (image processing)
- Function blocks [► 1281]
  (communication with cameras and the file system)

6.1 Software concept

This chapter describes the innovations or special features that TwinCAT Vision brings along. The principles of the PLC programming in TwinCAT 3 as well as the reference programming can be found in the general TwinCAT 3 documentation. In the reference programming you will find a chapter with TwinCAT 3 programming conventions. The recommendations and notes as well as the advantages and disadvantages that it contains enable a standardized program structure, a consistent naming of objects, variables, instances and an easily readable and more understandable code to be achieved and thus the simplification of the development, use and maintenance of the programs.

Unambiguous access to library modules

If the Tc3_Vision library is to be included (referenced) in another library and can thus occur several times in a project, an unambiguous access must be made via the namespace as prefix of the identifier. Default e.g.:

hr := Tc3_Vision.F_VN_TransformIntoDisplayableImage(ipImageIn, ipImageInDisp, hr);

For more information and setting options, see the PLC project settings and the library properties.

Access to the same data when using several tasks

If several tasks are to access the same image, for example, these accesses must be synchronized according to these guidelines. Otherwise there is the danger of an inconsistent data set, which could lead to unforeseeable consequences and program malfunctions.

6.1.1 Interface Pointers

Dynamic memory areas are required for TwinCAT Vision in order, for example, to save images and containers. These images and containers are represented by variables of type ITcVnImage [► 383] and ITcVnContainer [► 345]. Since they are interfaces pointing to a dynamic memory area, these variables are called interface pointers.

With the TwinCAT Vision API functions, the associated memory management is automatically controlled within the functions; nothing needs to be observed here.

For example, a new image is created (800 x 600 pixels of the type TCVN_ET_USINT with one channel):

VAR
  hr : HRESULT;
  ipImage : ITcVnImage;
END_VAR

hr := F_VN_CreateImage(ipImage, 800, 600, TCVN_ET_USINT, 1, hr);
this image is now to be converted into an RGB image with 3 channels. Therefore 3 times as much memory is required. The following function call is sufficient for this:

```c
hr := F_VN_ConvertColorSpace(ipImage, ipImage, TCVN_CST_GRAY_TO_RGB, hr);
```

Internally it frees up the previous memory area and allocates new memory in a size to suit. Interface pointers are thus a simple way of managing objects with dynamically required memory in the PLC.

⚠️ WARNING

**Consequences of misuse**

If you do not follow the following instructions for the use of interface pointers, this can lead to **complete system crashes** and to **memory leaks**!

---

**Reference counter**

As soon as the variables are no longer exclusively used as function parameters for the TwinCAT Vision API functions, however, it is necessary to familiarize yourself more precisely with the concept of interface pointers.

It is important to know that the interface pointer variable is only a pointer to an object in memory. If you assign one variable to another, only the pointer to this memory area is copied, not the data itself.

This means that if the data in one of the two variables is changed, it is also automatically changed in the other.

If the memory in one of the variables is released (as in the above example `F_VN_ConvertColorSpace`), and the second variable still has its pointer to this memory area, accessing this already released memory tends to result in a system crash.

For this reason, the objects in the memory all have a reference counter. As soon as a variable is assigned to another, this reference counter must be increased. If a new pointer is assigned to a variable, the reference counter must be decreased again.

When the reference counter reaches 0, the memory is released. The corresponding data are thus completely deleted.

In order to indicate whether a pointer variable points to a valid memory area, this pointer must always be set to 0 if it is no longer in use (and the reference counter was thus reduced). By default, all created variables (and thus the pointers too) in the PLC are initially set to 0 when the program starts.

**Management of interface pointers**

In the TwinCAT Vision API functions (marked by `F_VN_...`), the reference counter is automatically managed by interface pointers. Thus, nothing special needs to be observed when using them. However, if interface pointers are manually copied or used, for example, as function parameters, manual management is necessary (see below).

**Checking the validity of interface pointers**

Since the value 0 for an interface pointer basically corresponds to just one invalid memory area – and access to it also leads to a critical system error – it is important to check before each use whether the value is non-zero.

In the TwinCAT Vision API functions (`F_VN_...`), this check and the corresponding handling take place internally and automatically. However, this is not possible with interface methods, such as `TcRelease()` in the following sample, since their call already leads to an error if the variable is 0. The variable must therefore be checked explicitly beforehand.
**WARNING**

**Method access to invalid memory area leads to system crash**

An internal check is not possible for interface methods, since the corresponding objects must exist to call the method.

Even outside the method call, make sure that the corresponding object exists (the interface pointer should be non-zero).

In addition, a system crash does not necessarily always occur. If the invalid memory area coincidentally remains unused, a system crash may not occur.

---

**Enabling an interface pointer**

The enablement of an interface pointer variable consists of checking whether this is already 0 and otherwise reducing the reference counter and setting the variable to 0. As this is a frequent use case, a function `FW_SafeRelease` is provided that carries out all necessary steps with a single call:

```plaintext
hr := FW_SafeRelease(ADR(ipImage));
```

The internal mode of operation of the `FW_SafeRelease` function is illustrated by the following code snippet:

```plaintext
IF ipImage <> 0 THEN
    // Check if interface pointer is already = 0
    // If not, call .TcRelease() and set it to 0
    ipImage.TcRelease();
    // decrement reference count
    ipImage := 0;  // set to 0
END_IF
```

**Copying of an interface pointer**

When copying an interface pointer, its reference counter must be manually incremented. A direct assignment by means of the `:=` operator, as well as function/method/function block calls in which an interface pointer is transferred as an input, output or return value, count as a copying procedure. A transfer by means of I/O image also counts. The following method of incrementing the reference counter is available:

```plaintext
ipImage.TcAddRef();  // increment reference count
```

In addition, it must be ensured that the interface pointer to be written is enabled before copying. Otherwise memory leaks will occur. A manual procedure for copying an interface pointer could look like the following, for example:

```plaintext
IF ipSrcImage <> 0 THEN
    // check if source pointer is valid
    FW_SafeRelease(ADR(ipDestImage));  // release destination pointer
    ipDestImage := ipSrcImage;  // assign source pointer to destination pointer
    ipSrcImage.TcAddRef();  // increment reference count
END_IF
```

Please note that only the interface pointer, not the underlying image is copied.

**Rules of thumb**

Stick to the following rules of thumb when handling interface pointers.

It is irrelevant whether the interface pointers are (displayable) images, containers, iterators, etc. – they are mutually interchangeable in the samples.

**If you call a method on an interface pointer**

It must be ensured that the interface pointer is valid (check for `<> 0`):

```plaintext
IF ipImage <> 0 THEN
    ipImage.GetImageInfo(stImageInfo);
END_IF
```

In general it is recommended to use the alternative `F_VN_` function (if it exists) instead of an interface method, because the check will then be automatically performed internally:

```plaintext
hr := F_VN_GetImageInfo(ipImage, stImageInfo, hr);
```
If you carry out the check and the call of the interface method in the same expression, you must use
**AND_THEN** instead of **AND**. Only in that case will the rear part be executed if the front part returns **TRUE**:

```plaintext
IF ipIterator <> 0 AND_THEN ipIterator.CheckIfEnd() <> S_OK THEN
  ipIterator.GetContainer(ADR(ipElement));
END_IF
```

**Manually copy an interface pointer**

Make sure beforehand that the interface pointer to be written has been enabled: Increment the reference counter of the interface pointer after copying:

```plaintext
FW_SafeRelease(ADR(ipDestImage));
IF ipSrcImage <> 0 THEN
  ipDestImage := ipSrcImage;
  ipSrcImage.TcAddRef();
END_IF
```

**Declare interface pointer in VAR_INPUT**

Nothing needs to be observed for functions and methods. For programs and function blocks, the pointer in
the variables is retained after the call and should therefore be set to 0 at the end of the POU:

```plaintext
VAR_INPUT
  ipSrcImage: ITcVnImage;
END_VAR
```

This ensures that the memory area can no longer be accessed the next time it is called, since it could have
already been released outside. Increasing the reference counter within the POU is not a solution, since a
new assignment overwrites the previous pointer and therefore cannot be released, resulting in a memory
leak. If the interface pointer is to be kept, it should be copied to a local variable.

**Declare interface pointer as reference in VAR**

Release the interface pointer at the end of functions and methods. The interface pointer no longer exists
after ending the POU; therefore the reference counter must be decremented beforehand. For programs and
function blocks, the interface pointer should be released as soon as it is no longer required:

```plaintext
VAR
  ipImageWork: ITcVnImage;
END_VAR
```

```plaintext
//… functional code
hr := FW_SafeRelease(ADR(ipImageWork));
```

**Not recommended - declare interface pointer in VAR_OUTPUT**

The declaration of an interface pointer in VAR_OUTPUT is not recommended, since the use of outputs is
optional and thus memory leaks can occur.
In programs and function blocks, the pointer in the variables is retained even after the call, so that in the
event of changes outside, a released memory area can be accessed in the function block.
To prevent this, you must always ensure that the outputs are used and, in the case of programs and function
blocks, that the reference counter is incremented or the output variable is set to 0 after the output variable
has been copied.
This is very laborious and error-prone, since you have to take care yourself to implement correctly at every
point in the code. Therefore, transferring it as a reference in VAR_INPUT is recommended instead.

**NOTE**

**Memory leaks**

The use of an interface pointer in VAR_OUTPUT is not recommended. Failure to observe the above points
may result in memory leaks and system crashes.

**Declare interface pointer as reference in VAR_INPUT**

The transfer as a reference should be used if a change to the interface pointer (e.g. assigning a new image/
container) is to be passed to the outside within the POU for further use there:

```plaintext
VAR_INPUT
  ipDestImage: REFERENCE TO ITcVnImage;
END_VAR```
The sample Self-written functions [1440] explains the correct use of interface pointers in self-written functions and methods.

**Consequences of misuse**

If an interface pointer variable is copied, but the reference counter is not adjusted at the same time with TcAddRef(), there are two pointers; however, only one of them is known to the program. Therefore, the data to which both interface pointer variables point will then be deleted as soon as one of the two variables is released. The release can take place both by a manually executed FW_SafeRelease and by a TwinCAT Vision API function that rewrites the variable. If the remaining interface pointer variable is then accessed by a method call, the system crashes as described above.

If, contrary to this, interface pointer variables are rewritten without prior release (e.g. through a := assignment), this leads to **memory leaks**. This can occur, for example, if interface pointers are used as method variables and not released before ending the method. An indication for memory leaks can be found in the information window of the AMS router (see Router memory [52]). If the memory marked there as available decreases over time, this indicates a memory leak.

### 6.1.2 HRESULT

All TwinCAT Vision functions return an HRESULT after their execution. Its value indicates whether the execution was successful or not.

**SUCCESS codes**

A successful execution is indicated by a POSITIVE code. In hexadecimal notation the first digit is between 0 and 7.

```c
hr := E_VN_ReadQRCode(ipImageIn, ipDecodedData, S_OK);
```

Frequent SUCCESS codes are:

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>16#000</td>
<td>S_OK</td>
<td>Function was executed successfully</td>
</tr>
<tr>
<td>16#001</td>
<td>S_FALSE</td>
<td>Function was executed successfully but did not reach a complete result. (Occurs, for example, with the Code Reading functions if no code was found in the image.)</td>
</tr>
<tr>
<td>16#203</td>
<td>S_PENDING</td>
<td>Asynchronous method was started but there is no result yet (occurs at the first call of fbCameraControl.StartAcquisition(), for example; at the second call S_OK is returned, for example)</td>
</tr>
<tr>
<td>16#256</td>
<td>S_WATCHDOGTIMEOUT</td>
<td>Function was aborted by the watchdog.</td>
</tr>
</tbody>
</table>

**ERROR codes**

A failed execution is indicated by a NEGATIVE code, or in hexadecimal notation the first digit is >= 8. The final digits of the code correspond to the ADS Return Codes [1486].

**NOTE**

If an error code is returned, it means that all returns of the function are invalid. and must therefore not be used.
Common ERROR codes with the TwinCAT Vision API elements are:

<table>
<thead>
<tr>
<th>Hex</th>
<th>Dec</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>16#70A</td>
<td>1802</td>
<td>NOMEMORY</td>
<td>Insufficient memory</td>
</tr>
<tr>
<td>16#70B</td>
<td>1803</td>
<td>INVALIDPARM</td>
<td>Invalid parameter values</td>
</tr>
<tr>
<td>16#70C</td>
<td>1804</td>
<td>NOTFOUND</td>
<td>Not found (file, image, ...)</td>
</tr>
<tr>
<td>16#70E</td>
<td>1806</td>
<td>INCOMPATIBLE</td>
<td>Objects do not match</td>
</tr>
<tr>
<td>16#712</td>
<td>1810</td>
<td>INVALIDSTATE</td>
<td>The method of an FB was called in an impermissible state.</td>
</tr>
<tr>
<td>16#719</td>
<td>1817</td>
<td>TIMEOUT</td>
<td>timeout</td>
</tr>
<tr>
<td>16#71A</td>
<td>1818</td>
<td>NOINTERFACE</td>
<td>Interface query failed</td>
</tr>
<tr>
<td>16#71B</td>
<td>1819</td>
<td>INVALIDINTERFACE</td>
<td>Wrong interface requested.</td>
</tr>
<tr>
<td>16#71D</td>
<td>1821</td>
<td>INVALIDOBJID</td>
<td>Object ID is invalid.</td>
</tr>
<tr>
<td>16#734</td>
<td>1844</td>
<td>OUTOF RANGE</td>
<td>Outside the valid range.</td>
</tr>
</tbody>
</table>

Please refer to the ADS Return Codes [1486] for the complete list.

**Application**

In the event of an error, the HRESULT is negative (error code), otherwise it is positive (success code). Use the FAILED () and SUCCEEDED () functions to check this.

```plaintext
PROGRAM MAIN
VAR
  hr : HRESULT := S_OK;
END_VAR
IF SUCCEEDED(hr) THEN
  //code
END_IF
```

**Restricted query**

Note that the above query only checks for successful execution of the function, not for a correct or desired result.

For example, Code Reading functions [753] return the success code S_FALSE (16#001) if no code is found or read in the image. To detect this case, hr must be directly compared with S_OK or S_FALSE. In contrast, measurement functions [1215] return NOTFOUND (16#70C), if the object to be measured is missing from the image.

Most TwinCAT Vision functions expect a HRESULT as the last input parameter and are only executed if this is a SUCCESS code and it is therefore confirmed that no error has occurred so far. Otherwise, the respective function is not executed and instead returns the previous HRESULT. This ensures that, when the HRESULT is passed from function to function, the first error code that occurred is output instead of a subsequent error.

**SUCCESS codes**

SUCCESS codes (e.g. S_FALSE) are not forwarded and overwritten by the next function. If necessary, the HRESULT value of the respective functions must be checked explicitly.

**Accessing interface methods**

If a function is to create an interface pointer, the HRESULT of the function indicates whether the interface pointer now exists. In addition, the interface pointer must be checked to ascertain whether it is zero.

hr := fbCamera.GetCurrentImage(ipImageIn);
**WARNING**

**Method access to invalid memory area leads to system crash**

An internal check is not possible for interface methods, since the corresponding objects must exist to call the method.

Even outside the method call, make sure that the corresponding object exists (interface pointer not 0) and that the program has not yet returned an error code.

This is a secure query to use a newly created interface pointer:

```c
IF SUCCEEDED(hr) AND ipImage <> 0 THEN
    hr := ipImage.GetWidth(nWidth);
END_IF
```

**Integration into user-defined function blocks**

An **HRESULT** can be integrated into user-defined function blocks as follows:

```c
METHOD MyMethod : HRESULT
VAR_INPUT
    hr : HRESULT
END_VAR

IF FAILED(hr) THEN
    MyMethod := hr; //skip the whole method
ELSE
    (*
        code
        code
        code
    *)
    MyMethod := S_OK; // or some error code if something went wrong
END_IF
```

This is explained in more detail in the sample **Self-written functions [1440]**.

**Extraction of the ADS Return Code**

If, for example, the pure ADS Return Code is required for display in an HMI, this can be extracted from the **HRESULT** in numeric and textual form as follows:

```c
PROGRAM MAIN
VAR
    hr             :   HRESULT;
    nReturnCode    :   DWORD;
    sReturnCode    :   STRING;
END_VAR

nReturnCode := DINT_TO_DWORD(hr) AND 16#FFF;
sReturnCode := DWORD_TO_HEXSTR(nReturnCode, 3, FALSE);
```

The library Tc2_Utilities is required for this purpose.

### 6.1.3 Watchdogs

Since the runtime of many image processing algorithms depends on the image content, function calls can take much longer than usual under unfavorable circumstances.

This can be caused, for example, by changing lighting conditions, unexpected objects in the image and other factors that can drastically influence the calculation time of functions such as `F_VN_FindContours()` [1037]. If, for example, only 10 contours can be found under normal conditions, with other lighting conditions there may be 100 or more.

This can lead to cycle overruns, which must be avoided at all costs, as they lead to undefined behavior.
Watchdogs are helpful for this purpose. Watchdogs can abort individual Vision functions or entire code sections with several functions after a certain time. Some TwinCAT Vision functions can be aborted during execution and return the partial results calculated up to that point. If several functions are monitored by a watchdog, all remaining functions within the watchdog range are skipped when the specified time has elapsed. However, the subsequent functions (after the watchdog area) are executed normally again.

There are two functions available to define the start of a watchdog-monitored area, which differ in terms of when the specified time starts to count:

- **F_VN_StartAbsWatchdog** defines an absolute abort time relative to the start of the task cycle
- **F_VN_StartRelWatchdog** defines a relative abort time relative to the start of the watchdog area

In both cases, the end of the area is defined by the function **F_VN_StopWatchdog**, which optionally returns the handling component and the required time.

**NOTE**

Watchdog with relative abort time

When using **F_VN_StartRelWatchdog**, care must be taken that the abort time is not set too long, as otherwise cycle overshoots can continue to occur. Therefore, observe the cycle time of the corresponding task, the time already elapsed up to the call of the function and the time still required until the end of the task after **F_VN_StopWatchdog**.

In order for functions to be monitored by a watchdog, the option "Watchdog stack" must be enabled on the respective executing task.

Sample

In the sample below, a watchdog is started with a stop time of 10 ms relative to the call of the watchdog function. If, for example, the cycle time is 20 ms, 4 ms have already passed in the current task cycle when **F_VN_StartRelWatchdog** is called and the watchdog is started with a stop time tStop of 10 ms relative to the current time, then the watchdog stops the monitored functions (i.e. any function called between the start and stop of the watchdog) after 14 ms of the current cycle have elapsed.

```Var
ipImage : ITcVnImage := 0;
ipContours : ITcVnContainer := 0;
```
// watchdog runtime info
nFunctionsMonitored : UUINT;
nFractionProcessed : UDINT;
tRest : DINT;
END_VAR

/* imagine some other functions that use 4ms up to here */

hr := F_VN_StartRelWatchdog(10000, hr); // 10ms
    hr := F_VN_Threshold(ipImage, ipImage, 120, 255, TCVN_TT_BINARY, hr);
    hr := F_VN_FindContours(ipImage, ipContours, hr);
hk := F_VN_StopWatchdog(nFunctionsMonitored => nFunctionsMonitored, nFractionProcessed => nFractionProcessed, tRest => tRest);

Now two situations are possible:

• Both functions are terminated in time – this is the normal case.

  Suppose F_VN_Threshold [1166] takes 1 ms and F_VN_FindContours [1037] takes 5 ms. When
  F_VN_StopWatchdog [882] is called, 4 ms remain. The watchdog runtime information is:

  nFunctionsMonitored = 2
  nFractionProcessed = 100 // in %
  tRest = 4000 // in us, equals 4 ms

• The watchdog must intervene.

  Scenario: The function F_VN_Threshold [1166] is not dependent on the image content (only on the number
  of pixels) and therefore only takes 1 ms, but the lighting conditions have changed unfavorably, so that
  F_VN_FindContours would take longer than 9 ms. Therefore the watchdog stops F_VN_FindContours [1037],
  which nevertheless returns the contours found so far. The watchdog runtime information could then be as
  follows:

  nFunctionsMonitored = 2
  nFractionProcessed = 70
  tRest = -50

In this case, an estimated handling percentage of 70% was calculated when the function was aborted. The
remaining time is negative, i.e. the planned stop time was exceeded by 50 us, so that after the function
F_VN_StopWatchdog [882] 14050 us have passed since the start of the task cycle, instead of the planned
14000 us. This overrun is due to the fact that the partial results of the function that have already been
calculated are to continue to be used. Therefore, on the one hand the algorithm can only be aborted at
specific points, on the other hand the previous results must be organized and returned. The maximum
overrun generally depends on the specific function and the image content. In a program, the termination time
should therefore always be selected such that a safety buffer remains at the end of the task cycle.

Monitored functions

The following functions provide partial results in the event of a watchdog termination:

• F_VN_AdaptiveThreshold(Exp) [1162]
• F_VN_ConvertColorSpace [1083]
• F_VN_DetectBlobs(Exp) [1029]
• F_VN_FindContourHierarchyExp [1033]
• F_VN_FindContours(Exp) [1037]
• F_VN_HoughCircle(Exp) [1042]
• F_VN_LocateCircularArc(Exp) [1224]
• F_VN_LocateEdge(Exp) [1230]
• F_VN_LocateEdges(Exp) [1236]
• F_VN_LocateEllipse(Exp) [1242]
• F_VN_MatchTemplate(Exp) [1050]
API reference

- `F_VN_MatchTemplateAndEvaluate(Exp)` [1051]
- `F_VN_MeasureAngleBetweenEdges(Exp)` [1248]
- `F_VN_MeasureEdgeDistance(Exp)` [1253]
- `F_VN_MeasureMinEdgeDistance(Exp)` [1258]
- `F_VN_NormalizeImageForDisplay()` [1096]
- `F_VN_ReadDataMatrixCode(Exp)` [759]
- `F_VN_ReferenceColorSimilarity(Exp)` ITcVnColorModel (ITcVnMlModel) [1097]
- `F_VN_ReferenceColorSimilarity(Exp)` TcVnVector3 LREAL [1099]
- `F_VN_ResizeImage()` [1015]
- `F_VN_Threshold()` [1166]
- `F_VN_TrainImageColor(Exp)` [1102]
- `F_VN_WarpAffine(Exp)` [1017]
- `F_VN_WarpPerspective(Exp)` [1021]

Other functions cannot be aborted during the execution, but will be skipped if the watchdog time has expired.

Examples
- Blob Detection with watchdog monitoring [1390]

6.1.4 Image processing

In the following chapters, some concepts of the image processing are explained both generally and specifically with reference to TwinCAT Vision. They will help you to understand the mode of operation of many API functions [401].

6.1.4.1 Images

Images are handled in TwinCAT Vision by image objects, which could have implemented various interfaces such as ITcVnImage [383]. Access to the actual image data of the image object is gained via an interface pointer, which is then used in the PLC for processing. The images can consist of various pixel types and a number of channels, and may have a virtually arbitrary size. Due to processing from the real-time and the dynamic size, the image data are located in the router memory. Camera images are stored there directly after receipt, whereas images from the hard disk must first be loaded via a File Source or PLC function block and end up in the router memory in this way. The assistants in the camera object do not work with images from the router memory.

Querying an image interface of a Vision device

In many cases images are created with a camera or by loading an image file. The following function blocks are available for this purpose:

- `FB_VN_SimpleCameraControl` [1348]
  Can be connected to a GigE Vision camera instance or a File Source instance.
- `FB_VN_GevCameraControl` [1327]
  Is connected to a GigE Vision camera instance.
- `FB_VN_FileSourceControl` [1316]
  Is connected to a File Source instance.

Depending on the setting of the Vision device, a startup process or an image trigger may be necessary. Only the querying of the interface pointer in the PLC, via which access is gained to the image in the memory, is displayed here:
VAR 
   hr: HRESULT;
   fbVisionDevice: FB_VN_SimpleCameraControl;
END_VAR

hr := fbVisionDevice.GetCurrentImage(ipImageIn);
IF SUCCEEDED(hr) AND ipImageIn <> 0 THEN
   // Process image...
END_IF

Creating images

There are situations where an image is needed that has not been captured by a camera or loaded from the file system. For example, a color image might be created to display results. The function F_VN_CreateImage can be used for this purpose by specifying the image size, pixel type and number of channels. The pixel type TCVN_ET_USINT selected here corresponds to a usual image with 8 bits per channel. Please note that the memory required for the image is only allocated, but not initialized. Therefore the image can be set to a uniform color with the function F_VN_SetPixels.

Maximum image size

The maximum image size that can be created, received, or processed is limited by the number of 2^20 pixels for rows or columns, or by the total size of 2^30 pixels for an image, whichever comes first.

PROGRAM MAIN
VAR
   ipImage : ITCvImage;
   aColorBlack : TcvnVector4_LREAL := [0, 0, 0];
   hr : HRESULT;
END_VAR

hr := F_VN_CreateImage(
   ipImage := ipImage,
   nWidth := 640,
   nHeight := 480,
   ePixelType := TCVN_ET_USINT,
   nChannelNum := 3,
   hrPrev := hr);
hr := F_VN_SetPixels(ipImage, aColorBlack, hr);

Alternatively, an image can be created from existing image data using the F_VN_CreateImageFromArray function.

PROGRAM MAIN
VAR
   aImageData : ARRAY [0..479, 0..639] OF USINT;
   ipImage : ITCvImage;
   hr : HRESULT;
END_VAR

hr := F_VN_CreateImageFromArray(
   pData := ADR(aImageData),
   ipImage := ipImage,
   nWidth := 640,
   nHeight := 480,
   ePixelType := TCVN_ET_USINT,
   nChannelNum := 1,
   hrPrev := hr);

Accessing image data

The functions F_VN_GetPixel and F_VN_SetPixel can be used to access individual pixels of an image. This method is easy to use. However, it has a relatively long runtime when applied repeatedly to many pixels. In this case, direct access to the image data array is more appropriate. A pointer for a complete row of pixels is fetched and the array operator is used to access the individual pixels:

PROGRAM MAIN
VAR
   ipImage : ITCvImage;
   nHeight : UDINT;
   nWidth : UDINT;
END_VAR
Displayable images

In order to be able to transfer and display an image by ADS (e.g. in the ADS Image Watch [125]), it must first be converted into a displayable image of the type ITcVnDisplayableImage [383].

The strict distinction between displaying and processing images is made for the following reason: Image transmission via ADS is not synchronous with the PLC cycle. Consequently, image processing functions could be performed at the same time as image transfer. In order to avoid memory conflicts during image transfer, it is therefore necessary to separate the memory areas from the images to be displayed and processed.

The two interfaces ITcVnImage [383] and ITcVnDisplayableImage [383] prevent image access for unwanted purposes (either processing or transferring). The following functions are available for conversion into a displayable image (ITcVnDisplayableImage [383]):

- F_VN_CopyIntoDisplayableImage [721]
- F_VN_TransformIntoDisplayableImage [750]
- F_VN_TransformIntoDisplayableImageExp [751]

The F_VN_CopyIntoDisplayableImage [721] function creates a deep copy of the image data and makes it available as a displayable image. It should always be used if the image to be displayed is to be used in the following program sequence:

```plaintext
// Processing, e.g.:
hr := F_VN_Threshold(ipImage, ipImage, 128, 255, TCVN_TT_BINARY, hr);

// Show intermediate processing step
hr := F_VN_CopyIntoDisplayableImage(ipImage, ipImageDisp, hr);

// Further processing, e.g.:
hr := F_VN_FindContours(ipImage, ipContours, hr);
```

The function F_VN_TransformIntoDisplayableImage [750] uses the existing image data and releases the original interface pointer of the image. This means that the image data can no longer be processed and are fully available for image transmission. Since no copy of the image data is created, the execution time of this function is much shorter and no additional memory is used. However, this function only works if there is no other pointer to the image data. Otherwise the error code E_INCOMPATIBLE (16#70E) is returned. This can occur among other things if the Ads Communicator object [118] is activated (e.g. stream recording [98]), as this continues to reference the image internally for a short time. This function is therefore mainly useful for the end of the processing chain in order, for example, to display an already generated result image.
// Put last results on image, e.g.:
hr := F_VN_DrawContours(ipContours, -1, ipImageRes, aColorRed, 5, hr);

// Finally transform result image into a displayable image
hr := F_VN_TransformIntoDisplayableImage(ipImageRes, ipImageResDisp, hr);

In addition, the function F_VN_TransformIntoDisplayableImageExp [751] offers the option to create a deep copy of the original image on demand. Thus, the resource-saving transform variant is used if possible, while the copy variant is used if necessary. A corresponding use case can occur if the AdsCommunitor object [117] of a GigE Vision Camera [66] instance is activated to save images as a stream to the file system. In addition to the pointer in the PLC, there is another image data pointer in the AdsCommunicator object. If this is not yet enabled in the PLC at the time the image is used, the image must be copied for display.

hr := fbCamera.GetCurrentImage(ipImageIn);
hr := F_VN_CopyImage(ipImageIn, ipImageWork, hr);

// Reliably show original image and try to reuse image data to avoid deep copy
hr := F_VN_TransformIntoDisplayableImageExp(
    ipSrcImage := ipImageIn,
    ipDestImage := ipImageInDisp,
    bAllowDeepCopy := TRUE,
    hrPrev := hr
);

// Manipulate ipImageWork during processing here ...

### Memory limitation for displayable images

Images that are displayed in the development environment (ADS Image Watch or camera assistants) are located in the same RAM area as the respective Visual Studio instance. If you now wish to view a very large number of images or images of a large size, bottlenecks may occur due to the maximum available memory per instance.

As the images are retrieved from the PLC for display in the ADS Image Watch (ADS communication) and this exchange takes place via the router memory, sufficient memory space must be available at this point.

### Meta information

Meta information for an image can be obtained with the function F_VN_GetImageInfo [737]. As a result of the function, a structure of the type TcVnImageInfo [207] is returned. The information it contains can be used, for example, to ensure that an image has the expected format in order to be converted:

hr := F_VN_GetImageInfo(ipImageIn, stImageInfo, hr);
IF stImageInfo.stPixelFormat.nChannels = 3 THEN
    hr := F_VN_ConvertColorSpace(ipImageIn, ipImageIn, TCVN_CST_RGB_TO_GRAY, hr);
END_IF

### 6.1.4.2 Container

A typical task in image processing is to find objects in the image and to determine their shape, position and size. The objects found can be described, for example, by their contour points. Since the number of contour points depends on the specific object, it can vary greatly. For this reason, containers of dynamic size are used to store these contours.

However, the containers are not limited to storing contour points, they can also contain a variety of other data types. Different data types cannot be mixed within a container. Some basic concepts for handling containers are explained below.

#### Creating a container

A container can be created with a single function call. Only the type (see available types [151]) and the initial number of elements have to be defined:

```pascal
PROGRAM MAIN
VAR
    hr : HRESULT;
    ipContainer : ITcVnContainer;
```
In this sample a new container with 10 elements of type `<REAL>` was created, which is internally organized as C++ vector. Alternatively, the function `F_VN_CopyContainer` can be used to create a deep copy of an existing container. Not only the pointer to the data is copied - as with an assignment to another variable - but the data is copied to a newly allocated memory area.

All available types of containers are stored as global constants and named according to the following pattern: All names begin with "ContainerType_". This is followed by the type, which is composed of the description of the actual container type (based on C++ containers) and the element types (e.g. `<REAL>`, `<UDINT>`, etc.). Containers can in turn also contain further containers as elements. Each level of the container structure is represented in the name by the respective container type, e.g. "Vector_" or "String_". Therefore the resulting type name of a container with `<REAL>` elements is `ContainerType_Vector_REAL`.

Adding elements

Since the containers are able to dynamically expand the allocated memory, additional elements can be added to an existing container. The functions `F_VN_AppendToContainer` and `F_VN_InsertIntoContainer` can be used for this.

However, as it may be internally necessary to allocate a completely new memory area and to copy the existing data, it is recommended to reserve the maximum required memory in advance. Consequently, new memory is allocated only once and the program has an overall higher performance. A function exists for this, `F_VN_ReserveContainerMemory`, which merely reserves memory, but does not change the number of elements in the container.

```plaintext
hr := F_VN_ReserveContainerMemory(ipContainer, 100, hr);
```

```plaintext
hr := F_VN_AppendToContainer_REAL(1.23, ipContainer, hr);
```

```plaintext
hr := F_VN_InsertIntoContainer_REAL(4.56, ipContainer, 2, hr);
```

Accessing elements

In the case that only a few individual elements of the container are to be accessed, it is recommended to use the `F_VN_GetAt_...` or `F_VN_SetAt_...` functions. Since no function name overload is possible in the PLC, there is a separate function name for each supported container type. The first element in the container has the index 0.

```plaintext
PROGRAM MAIN
VAR
    hr : HRESULT;
    ipContainer : ITcVnContainer;
    fElement : REAL;
END_VAR

hr := F_VN_GetAt_REAL(ipContainer, fElement, 0, hr); // Get first element
hr := F_VN_SetAt_REAL(fElement, ipContainer, 2, hr); // Set third element
```

The advantage of using these ready-made functions is that only a single line of code is required and all checks (e.g. whether the container type is correct and the index is not too large) are performed internally. If you wish to access all elements in the container one after the other, it is recommended in particular with large containers to access the container elements via iterators and the associated access interface in order to achieve a better performance:

```plaintext
PROGRAM MAIN
VAR
    hr : HRESULT;
    ipContainer : ITcVnContainer;
    ipIterator : ITcVnForwardIterator;
    ipAccess : ITcVnAccess_REAL;
    fElement : REAL;
END_VAR

hr := F_VN_GetForwardIterator(ipContainer, ipIterator, hr);
IF SUCCEEDED(hr) AND ipIterator <> 0 THEN
```
hr := ipIterator.TcQueryInterface(IID_ITcVnAccess_REAL, ADR(ipAccess));
IF SUCCEEDED(hr) AND ipAccess <> 0 THEN
    WHILE SUCCEEDED(hr) AND ipIterator.CheckIfEnd() <> S_OK DO
        hr := ipAccess.Get(fElement);
        IF SUCCEEDED(hr) THEN
            fElement := fElement + 1;
            hr := ipAccess.Set(fElement);
        END_IF
    END_WHILE
END_IF
hr := FW_SafeRelease(ADR(ipAccess));
END_IF
hr := FW_SafeRelease(ADR(ipIterator));

If the elements in the container are in turn containers, use the following procedure instead.

**Container of containers**

Since, for example, the function `F_VN_FindContours` can find not only one but several contours, containers can contain further containers as elements (in which case all inner containers must contain elements of the same type). In principle, these can be used in exactly the same way as the simple containers described above. For this purpose there is, for example, the function `F_VN_GetAt_ITcVnContainer` for obtaining a deep copy of any inner container that is usable from the PLC (it is technically impossible to obtain an interface pointer to the original data). Due to the general `ITcVnContainer` data type at this level, it is not necessary to distinguish which base element types are contained in the inner containers. This simplifies access via iterators, since the special Access Interface is no longer necessary:

```pascal
PROGRAM MAIN
VAR
    hr : HRESULT;
    ipContainer : ITcVnContainer;
    ipIterator : ITcVnForwardIterator;
    ipElement  : ITcVnContainer;
END_VAR
hr := F_VN_GetForwardIterator(ipContainer, ipIterator, hr);
IF SUCCEEDED(hr) AND ipIterator <> 0 THEN
    WHILE SUCCEEDED(hr) AND ipIterator.CheckIfEnd() <> S_OK DO
        // ipElement gets a deep copy!
        hr := F_VN_GetContainer(ipIterator, ipElement, hr);
        IF SUCCEEDED(hr) AND ipElement <> 0 THEN
            // 1. extract information or manipulate ipElement
            // 2. write back changes if ipElement was manipulated
            hr := F_VN_SetContainer(ipIterator, ipElement, hr);
        END_IF
        hr := F_VN_IncrementIterator(ipIterator, hr);
    END_WHILE
END_IF
hr := FW_SafeRelease(ADR(ipElement));
END_IF
hr := FW_SafeRelease(ADR(ipIterator));
```

**Displaying containers**

At present the content of containers cannot be displayed in the Visual Studio Live Debugging without further action. Instead, the data can be exported to an array and viewed in this way:

```pascal
VAR
    aArray       : ARRAY [0..9] OF REAL;
    ipContainer  : ITcVnContainer;
    nBufferSize  : ULINT;
END_VAR
hr := F_VN_ExportContainerSize(ipContainer, nBufferSize, hr);
IF nBufferSize = SIZEOF(aArray) THEN
    hr := F_VN_ExportContainer(
        ipContainer :=ipContainer,
        pBuffer     :=ADR(aArray),
        nBufferSize :=nBufferSize,
        hr
    );
END_IF
```
Overwriting containers
Generally, containers with API functions can simply be overwritten. At present, however, overwriting is not possible if the existing container and the container to be written are of different types. This is signaled by the error code 70E (INCOMPATIBLE).

6.1.4.3 Contours

This chapter describes the use of contours in TwinCAT Vision.

What are contours?
In general, a contour (also called an outline) is the demarcation of an object from its surroundings; this is classically two-dimensional in image processing. A contour describes the shape, size and location of this object. Within the scope of image processing, this means an object in an image; hence, a contour describes the outline of the object in the image.

How are contours displayed?
A contour is displayed as a collection of 2D points. The points mostly refer to the origin of the image in which the object is mapped, provided no ROI has been set. If you connect all the points in this field with straight lines, you get a drawing of the contour.

How are contours displayed in TwinCAT Vision?
A dynamic array of 2D points is handled in TwinCAT Vision as a container of points (TcVnPoint2_DINT/REAL). A corresponding container then has the type designation ContainterType_Vector_TcVnPoint2_DINT/REAL.

The points then frequently lie directly on the pixel grid (e.g. in case of a contour search). In this case, DINT elements are used to save the points. However, there are also cases in which the accuracy is insufficient, and REALs therefore have to be used (e.g. in case of measurements). The points then do not lie exactly at the center point of a pixel, but somewhere else.

Drawing functions can only draw contours with integer elements in an image. In case of doubt, therefore, the element type of the container must be converted.
Fields of contours

A frequent occurrence is where several contours are to be handled at the same time (e.g. several contours are usually found with F_VN_FindContours). This is technically implemented by using a two-dimensional container. Therefore, there is an array of contours or an array of arrays of 2D points. The corresponding type designation is then ContainerType_Vector_Vector_TcVnPoint2_DINT/REAL.

What is contour approximation?

Generally, a very large number of points can always be used for displaying an object. However, this is frequently unnecessary, as (approximately) straight lines can be approximated through their start and end point. The points in between do not need to be saved in addition, because the start and end point suffice for the definition.

Depending on the method used and the geometry of the objects, the approximation can be more or less lossy. In any case, in the case of differing approximation, the geometrical features can also be calculated with slightly different values.

Geometrical features

On contours, i.e. from the arrays of points, a large number of individual features can be calculated that geometrically describe the respective contour. The contour and thus the respective object can be examined and, for example, classified via these features. These features include:

- Scope
- Surface
- Center points
Closed and open contours

A contour is normally closed. In this case the first and last element of the array are simply regarded as being connected. However, it is also possible to have an open contour. Open and closed contours thus do not differ in terms of their technical representation, but by the information regarding how the data are to be interpreted. Whether the respective contour is to be used open or closed will be queried accordingly at the relevant points of the API.

Samples

- Object Detection [1390]
- Contour analysis [1380]

6.1.4.4 Masks

This chapter describes the meaning of masks in TwinCAT Vision.

What is a mask?

In relation to image processing, a mask means a mask image that belongs to another image [140]. The mask image has the same size and consists of a grayscale channel. The intensities of the mask image define which pixels of the associated image are to be taken into account for certain work steps.

What is a mask needed for?

Without further action, images are limited to a rectangular shape. If shapes other than rectangles are to be defined during image operations, this can be implemented by means of a mask. The mask is displayed by a further image with the same shape and size on which the areas (pixels) to be taken into account are marked differently.

How is a mask technically displayed?

The intensities of the mask grayscale image describe which pixels are to be considered. The interpretation of the intensities can in principle deviate from function to function. Binarily, it is frequently the case that all pixels with a value of 0 are interpreted as not to be considered and all pixels with a different value (>=1) are interpreted as to be considered. However, cases are also conceivable in which the degree of consideration can be set with a finer granularity.

A mask that restricts the observation area to a circular area in the image can look like this, for example:

![Circular Mask Example](image)

When applying this mask, the pixels in the corner of an associated image have no further influence on a subsequent operation.
Sample

By way of example, a mask is created for the image \texttt{ipImageWork} that says that only a circular area of the image is to be considered:

\begin{verbatim}
hr := F_VN_GetImageWidth(ipImageWork, nWidth, hr);
hr := F_VN_GetImageHeight(ipImageWork, nHeight, hr);

hr := F_VN_CreateImage(ipImageMask, nWidth, nHeight, TCVN_ET_USINT, 1, hr);
hr := F_VN_SetPixels(ipImageMask, aColorBlack, hr);
hr := F_VN_DrawCircle(nWidth/2, nHeight/2, MIN(nWidth, nHeight)/2, ipImageMask, aColorWhite, -1, hr);
\end{verbatim}

The average intensity value of the image within the mask area is subsequently calculated:

\begin{verbatim}
hr := F_VN_ImageAverageExp(ipImageWork, aAverage, ipImageMask, hr);
fAverageIntensityInCircle := aAverage[0];
\end{verbatim}

Without a mask it would not be possible without further action to restrict the average value to the circular area instead of the complete rectangular image.

The full sample can be found here: Average intensity in shapes of any kind [\texttt{1405}].

6.1.4.5 Region of Interest

This chapter describes the use of a Region of Interest in TwinCAT Vision.

What is a Region of Interest?

A Region of Interest (ROI) is a rectangular image region to which an image is temporarily restricted. The image itself is not changed, but the section to be processed merely reduced in size. An ROI can be reset to the entire image at any time.

What is a Region of Interest needed for?

An existing image is frequently larger than the area to be processed. In this case an ROI should be set in order to obtain the following advantages:

1. Better performance: the image processing time becomes shorter if the image region to be processed is smaller.
2. Fewer disruptions: disruptive image objects located outside the ROI are excluded from the processing and therefore do not influence the result.

It is equally possible through the use of ROIs to process several image areas separately by setting the ROI at various points in the image, one after the other.

\section*{NOTE}

ROI configuration in the camera

If you use a camera as an image source, it is best to implement an ROI there in order to reduce the data quantity to be transmitted and to keep it as low as possible. Depending on the setting and sensor, the possible frame rate may also increase as a result of this. See camera configuration examples: Region of Interest (ROI) [\texttt{1455}]

How is a Region of Interest technically displayed?

The information about a set ROI is stored in TwinCAT Vision directly in the respective image object. The setting of an ROI merely consists of the setting of a rectangle description. The image data outside of this area still exist, but they are ignored when calling functions.

The rectangle description consists of the position (\texttt{nX} and \texttt{nY}) of the top left corner and the size (\texttt{nWidth} and \texttt{nHeight}) of the rectangle. Make sure that the rectangle does not extend beyond the border of the image (\texttt{nWidth} \leq \texttt{ImageWidth} - \texttt{nX} and \texttt{nHeight} \leq \texttt{ImageHeight} - \texttt{nY}).
To set an ROI, these parameters are transferred directly or as a structure of the type `TcVnRectangle_UDINT` depending on the function:

- `F_VN_SetRoi`  
- `F_VN_SetRoi_TcVnRectangle_UDINT`

The function `F_VN_GetRoi` is available for checking whether and which ROI is set on an image. An ROI can also be set on an image that already has an ROI. The function `F_VN_ResetRoi` can be used if the ROI is to be enlarged or reset to the entire image size.

### 6.2 Data Types

There are three data types, each with the prefix "TcVn":

- **Alias names** for certain array types refer to image processing
- ** Enums** help with the parameterization of functions and contain the additional prefix "E"
- ** Structures** consolidate several related pieces of information (e.g. for geometric descriptions)

In addition, ** constants** are available for the description of container types.
6.2.1 Aliases

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>GVCP_REGISTER_ADDRESS</td>
<td>UDINT</td>
</tr>
<tr>
<td>TcVnPoint</td>
<td>TcVnPoint2_DINT</td>
</tr>
<tr>
<td>TcVnPoint2_DINT</td>
<td>TcVnVector2_DINT</td>
</tr>
<tr>
<td>TcVnPoint2_LREAL</td>
<td>TcVnVector2_LREAL</td>
</tr>
<tr>
<td>TcVnPoint2_REAL</td>
<td>TcVnVector2_REAL</td>
</tr>
<tr>
<td>TcVnPoint3_LREAL</td>
<td>TcVnVector3_LREAL</td>
</tr>
<tr>
<td>TcVnPoint3_REAL</td>
<td>TcVnVector3_REAL</td>
</tr>
<tr>
<td>TcVnRectangle</td>
<td>TcVnRectangle_DINT</td>
</tr>
</tbody>
</table>

6.2.2 Constants

The following constants represent types of containers as GUID. These are required, for example, with functions such as F_VN_CreateContainer.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ContainerType_Vector_SINT</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_USINT</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_INT</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_UINT</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_DINT</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_UDINT</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_ULINT</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_REAL</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_LREAL</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnPoint2_DINT</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnPoint2_REAL</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnPoint2_LREAL</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnPoint3_REAL</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnPoint3_LREAL</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnPoint2_DINT</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnPoint2_REAL</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnPoint3_REAL</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnPoint3_LREAL</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnVector2_SINT</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnVector2_USINT</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnVector2_INT</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnVector2_UINT</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnVector2_DINT</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnVector2_REAL</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnVector3_SINT</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnVector3_UINT</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnVector3_REAL</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnVector3_USINT</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnVector3_INT</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnVector3_UINT</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnVector3_REAL</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnVector4_SINT</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnVector4_USINT</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnVector4_INT</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnVector4_UINT</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnVector4_DINT</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnVector4_LREAL</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnRectangle_DINT</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnKeyPoint</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnDMatch</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnDMatch</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_TcVnDMatch</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_String_SINT</td>
<td>GUID</td>
</tr>
<tr>
<td>ContainerType_Vector_String_SINT</td>
<td>GUID</td>
</tr>
</tbody>
</table>
### 6.2.3 Arrays

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcVnArray10_ITcVnImage</td>
<td>array [0..9] of PVOID</td>
</tr>
<tr>
<td>TcVnArray3_Point2_REAL</td>
<td>array [0..2] of TcVnPoint2_REAL [151]</td>
</tr>
<tr>
<td>TcVnArray4_LREAL</td>
<td>array [0..3] of LREAL</td>
</tr>
<tr>
<td>TcVnArray4_Point2_REAL</td>
<td>array [0..3] of TcVnPoint2_REAL [151]</td>
</tr>
<tr>
<td>TcVnArray7_LREAL</td>
<td>array [0..6] of LREAL</td>
</tr>
<tr>
<td>TcVnArray8_LREAL</td>
<td>array [0..7] of LREAL</td>
</tr>
<tr>
<td>TcVnMatrix2x3_LREAL</td>
<td>array [0..1, 0..2] of LREAL</td>
</tr>
<tr>
<td>TcVnMatrix3x3_LREAL</td>
<td>array [0..2, 0..2] of LREAL</td>
</tr>
<tr>
<td>TcVnVector2_DINT</td>
<td>array [0..1] of DINT</td>
</tr>
<tr>
<td>TcVnVector2_INT</td>
<td>array [0..1] of INT</td>
</tr>
<tr>
<td>TcVnVector2_LREAL</td>
<td>array [0..1] of LREAL</td>
</tr>
<tr>
<td>TcVnVector2_REAL</td>
<td>array [0..1] of REAL</td>
</tr>
<tr>
<td>TcVnVector2_SINT</td>
<td>array [0..1] of SINT</td>
</tr>
<tr>
<td>TcVnVector2_UINT</td>
<td>array [0..1] of UINT</td>
</tr>
<tr>
<td>TcVnVector2_USINT</td>
<td>array [0..1] of USINT</td>
</tr>
<tr>
<td>TcVnVector3_INT</td>
<td>array [0..2] of INT</td>
</tr>
<tr>
<td>TcVnVector3_LREAL</td>
<td>array [0..2] of LREAL</td>
</tr>
<tr>
<td>TcVnVector3_REAL</td>
<td>array [0..2] of REAL</td>
</tr>
<tr>
<td>TcVnVector3_SINT</td>
<td>array [0..2] of SINT</td>
</tr>
<tr>
<td>TcVnVector3_UINT</td>
<td>array [0..2] of UINT</td>
</tr>
<tr>
<td>TcVnVector3_USINT</td>
<td>array [0..2] of USINT</td>
</tr>
<tr>
<td>TcVnVector4_DINT</td>
<td>array [0..3] of DINT</td>
</tr>
<tr>
<td>TcVnVector4_INT</td>
<td>array [0..3] of INT</td>
</tr>
<tr>
<td>TcVnVector4_LREAL</td>
<td>array [0..3] of LREAL</td>
</tr>
<tr>
<td>TcVnVector4_REAL</td>
<td>array [0..3] of REAL</td>
</tr>
<tr>
<td>TcVnVector4_SINT</td>
<td>array [0..3] of SINT</td>
</tr>
<tr>
<td>TcVnVector4_UINT</td>
<td>array [0..3] of UINT</td>
</tr>
<tr>
<td>TcVnVector4_USINT</td>
<td>array [0..3] of USINT</td>
</tr>
</tbody>
</table>

### 6.2.4 Enums

#### 6.2.4.1 ETcVn2dCodeSearchStrategy

Offers search strategies for 2d code reading (multiple TCVN_CSS_XXX_INVERTED or multiple TCVN_CSS_XXX_FLIPPED cannot be combined).

**Syntax**

**Definition:**

```plaintext
class ETcVn2dCodeSearchStrategy {
    TCVN_CSS_DEFAULT := 1,
    TCVN_CSS_ONLY_NOT_INVERTED := 8,
    TCVN_CSS_FIRST_NOT_INVERTED := 10,
    TCVN_CSS_ONLY_INVERTED := 12,
    TCVN_CSS_FIRST_INVERTED := 14,
    TCVN_CSS_ONLY_NOT_FLIPPED := 64,
    TCVN_CSS_FIRST_NOT_FLIPPED := 80,
    TCVN_CSS_ONLY_FLIPPED := 96,
    TCVN_CSS_FIRST_FLIPPED := 112
} UDINT;
```

END_TYPE
### Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_CSS_DEFAULT</td>
<td>The algorithm searches for codes using the default searching strategy, all other flags are ignored (dependant on code type).</td>
</tr>
<tr>
<td>TCVN_CSS_ONLY_NOT_INVERTED</td>
<td>The algorithm searches for codes only in the non-inverted image.</td>
</tr>
<tr>
<td>TCVN_CSS_FIRST_NOT_INVERTED</td>
<td>The algorithm first searches for codes in the non-inverted and then in the inverted image.</td>
</tr>
<tr>
<td>TCVN_CSS_ONLY_INVERTED</td>
<td>The algorithm searches for codes only in the inverted image.</td>
</tr>
<tr>
<td>TCVN_CSS_FIRST_INVERTED</td>
<td>The algorithm first searches for codes in the inverted and then in the non-inverted image.</td>
</tr>
<tr>
<td>TCVN_CSS_ONLY_NOT_FLIPPED</td>
<td>The algorithm searches for codes only in the non-mirrored image.</td>
</tr>
<tr>
<td>TCVN_CSS_FIRST_NOT_FLIPPED</td>
<td>The algorithm first searches for codes in the non-mirrored and then in the mirrored image.</td>
</tr>
<tr>
<td>TCVN_CSS_ONLY_FLIPPED</td>
<td>The algorithm searches for codes only in the mirrored image.</td>
</tr>
<tr>
<td>TCVN_CSS_FIRST_FLIPPED</td>
<td>The algorithm first searches for codes in the mirrored and then in the non-mirrored image.</td>
</tr>
</tbody>
</table>

**Further information**

This enum provides setting options for the search strategy of code reading functions. For more information see [Code reading search strategies](#754).

**Related functions**

- [F_VN_ReadDataMatrixCodeExp](#761)
- [F_VN_ReadQRCodeExp](#768)

### 6.2.4.2 ETcVnAdaptiveThresholdMethod

Offers methods for adaptive threshold.

#### Syntax

Definition:

```pascal
TYPE ETcVnAdaptiveThresholdMethod : (
  TCVN_ATM_MEAN := 0,
  TCVN_ATM_GAUSSIAN := 1
) DINT;
END_TYPE
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_ATM_MEAN</td>
<td>The threshold value is calculated as the mean of the nBlockSize x nBlockSize neighborhood of (x,y) minus fConstant.</td>
</tr>
<tr>
<td>TCVN_ATM_GAUSSIAN</td>
<td>The threshold value is the weighted sum (cross-correlation with a Gaussian window) of the nBlockSize x nBlockSize neighborhood of (x,y) minus fConstant.</td>
</tr>
</tbody>
</table>
6.2.4.3  ETcVnBarcodeSearchDirection

Offers search directions for linear barcodes.

Syntax

Definition:

```plaintext
TYPE ETcVnBarcodeSearchDirection :
{
   TCVN_BSD_ANY := 0,
   TCVN_BSD_HORIZONTAL := 1,
   TCVN_BSD_VERTICAL := 2
} UDINT;
END_TYPE
```

Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_BSD_ANY</td>
<td>The algorithm searches for codes in any supported directions.</td>
</tr>
<tr>
<td>TCVN_BSD_HORIZONTAL</td>
<td>The algorithm searches for codes in horizontal direction.</td>
</tr>
<tr>
<td>TCVN_BSD_VERTICAL</td>
<td>The algorithm searches for codes in vertical direction.</td>
</tr>
</tbody>
</table>

Related functions

- F_VN_ReadBarcodeExp [1163]

6.2.4.4  ETcVnBarcodeType

Offers linear barcode types to search for.

Syntax

Definition:

```plaintext
TYPE ETcVnBarcodeType :
{
   TCVN_BT_CODABAR := 4,
   TCVN_BT_CODE39 := 8,
   TCVN_BT_CODE93 := 16,
   TCVN_BT_CODE128 := 32,
   TCVN_BT_EAN8 := 128,
   TCVN_BT_EAN13 := 256,
   TCVN_BT_ITF := 512,
   TCVN_BT_UPCA := 32768,
   TCVN_BT_UPCE := 65536,
   TCVN_BT_ANY := 99260,
   TCVN_BT_CODE39EXTENDED := 262144
} UDINT;
END_TYPE
```

Related functions

- F_VN_ReadBarcodeExp [756]
### Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_BT_CODABAR</td>
<td>The algorithm searches for Codabar codes.</td>
</tr>
<tr>
<td>TCVN_BT_CODE93</td>
<td>The algorithm searches for Code-93 codes.</td>
</tr>
<tr>
<td>TCVN_BT_CODE128</td>
<td>The algorithm searches for Code-128 codes.</td>
</tr>
<tr>
<td>TCVN_BT_EAN8</td>
<td>The algorithm searches for EAN-8 codes.</td>
</tr>
<tr>
<td>TCVN_BT_EAN13</td>
<td>The algorithm searches for EAN-13 codes.</td>
</tr>
<tr>
<td>TCVN_BT_ITF</td>
<td>The algorithm searches for ITF codes.</td>
</tr>
<tr>
<td>TCVN_BT_UPCA</td>
<td>The algorithm searches for UPC-A codes.</td>
</tr>
<tr>
<td>TCVN_BT_UPCE</td>
<td>The algorithm searches for UPC-E codes.</td>
</tr>
<tr>
<td>TCVN_BT_ANY</td>
<td>The algorithm searches for any supported linear barcode. If the type of the code is known, it is recommended to select the specific type directly.</td>
</tr>
<tr>
<td>TCVN_BT_CODE39EXTENDED</td>
<td>The algorithm searches for Code-93-Extended codes.</td>
</tr>
</tbody>
</table>

### Related functions

- F_VN_ReadBarcode(Exp) [755]

### 6.2.4.5 ETcVnBlobCombination

Offers multiple blob contours to choose from, which should be returned in a multi-threshold scenario (used in TcVnParamsBlobDetection, which in return is used in F_VN_DetectBlobs).

#### Syntax

Definition:

```plaintext
TYPE ETcVnBlobCombination :
{
    TCVN_BC_SMALLEST := 0,
    TCVN_BC_LARGEST := 1,
    TCVN_BC_MIN_THRESHOLD := 2,
    TCVN_BC_MAX_THRESHOLD := 3,
    TCVN_BC_MEDIAN_THRESHOLD := 4
}DINT;
END_TYPE
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_BC_SMALLEST</td>
<td>Returns the smallest blob contour.</td>
</tr>
<tr>
<td>TCVN_BC_LARGEST</td>
<td>Returns the largest blob contour.</td>
</tr>
<tr>
<td>TCVN_BC_MIN_THRESHOLD</td>
<td>Returns the blob contour for the minimum threshold.</td>
</tr>
<tr>
<td>TCVN_BC_MAX_THRESHOLD</td>
<td>Returns the blob contour for the maximum threshold.</td>
</tr>
<tr>
<td>TCVN_BC_MEDIAN_THRESHOLD</td>
<td>Returns the blob contour for the median threshold.</td>
</tr>
</tbody>
</table>

### Further information

The enum ETcVnBlobCombination is used in the structure TcVnParamsBlobDetection [211].

### Related functions

- F_VN_CalibrateCamera(Exp) [944]
- F_VN_DetectBlobs(Exp) [1029]
6.2.4.6 ETCvNBorderInterpolationMethod

Offers methods to extrapolate values of non-existing pixels. On the one hand, this is used for filtering functions to enable filtering at the image borders (where the filter mask reaches over the border). On the other hand, this is used to extrapolate the undefined pixels after a geometric image transformation.

Syntax

Definition:

```
TYPE ETCvNBorderInterpolationMethod :
{
  TCVN_BIM_CONSTANT := 0,
  TCVN_BIM_REPLICATE := 1,
  TCVN_BIM_REFLECT := 2,
  TCVN_BIM_WRAP := 3,
  TCVN_BIM_REFLECT_101 := 4,
  TCVN_BIM_DEFAULT := 4,
  TCVN_BIM_TRANSPARENT := 5,
  TCVN_BIM_ISOLATED_CONSTANT := 16,
  TCVN_BIM_ISOLATED_REPLICATE := 17,
  TCVN_BIM_ISOLATED_REFLECT := 18,
  TCVN_BIM_ISOLATED_WRAP := 19,
  TCVN_BIM_ISOLATED_REFLECT_101 := 20
} DINT;
END_TYPE
```

Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_BIM_CONSTANT</td>
<td>iiiii</td>
</tr>
<tr>
<td>TCVN_BIM_REPLICATE</td>
<td>aaaaaa</td>
</tr>
<tr>
<td>TCVN_BIM_REFLECT</td>
<td>gfedcba</td>
</tr>
<tr>
<td>TCVN_BIM_WRAP</td>
<td>bcdedefgh</td>
</tr>
<tr>
<td>TCVN_BIM_REFLECT_101</td>
<td>hgfedcb</td>
</tr>
<tr>
<td>TCVN_BIM_DEFAULT</td>
<td>Choose this if you don't know which method to select (same as REFLECT_101).</td>
</tr>
<tr>
<td>TCVN_BIM_TRANSPARENT</td>
<td>Corresponding pixels in the destination image will not be modified. Only available for geometric image transformations.</td>
</tr>
<tr>
<td>TCVN_BIM_ISOLATED_CONSTANT</td>
<td>Similar to CONSTANT, but ROIs are handled as isolated, which means surrounding image pixels outside the ROI are ignored.</td>
</tr>
<tr>
<td>TCVN_BIM_ISOLATED_REPLICATE</td>
<td>Similar to REPLICATE, but ROIs are handled as isolated, which means surrounding image pixels outside the ROI are ignored.</td>
</tr>
<tr>
<td>TCVN_BIM_ISOLATED_REFLECT</td>
<td>Similar to REFLECT, but ROIs are handled as isolated, which means surrounding image pixels outside the ROI are ignored.</td>
</tr>
<tr>
<td>TCVN_BIM_ISOLATED_WRAP</td>
<td>Similar to WRAP, but ROIs are handled as isolated, which means surrounding image pixels outside the ROI are ignored.</td>
</tr>
<tr>
<td>TCVN_BIM_ISOLATED_REFLECT_101</td>
<td>Similar to REFLECT_101, but ROIs are handled as isolated, which means surrounding image pixels outside the ROI are ignored.</td>
</tr>
</tbody>
</table>

Further information

Border interpolation is used to simulate pixel values outside of the image. This is necessary, for example, when applying a filter or a morphological operator to an image. For example, if a filter iterates over the image, it must be specified how the filter behaves at the image edges where parts of the filter kernel lie outside of the image. For this purpose the non-existent pixels outside of the image are simulated so that the...
filter operation can still be performed. The method with which this simulation is performed is defined with the enum `ETcVnBorderInterpolationMethod`. The value `TCVN_BIM_REFLECT` is often a good choice; it reflects the outer image pixels with the image border as a mirror plane.

### Related functions

- `F_VN_BilateralFilterExp` [1115]
- `F_VN_CustomFilterExp` [1126]
- `F_VN_GaussianFilterExp` [1134]
- `F_VN_LaplacianFilterExp` [1138]
- `F_VN_PadImageBorderExp` [941]
- `F_VN_ScharrFilterExp` [1151]
- `F_VN_SeparableCustomFilterExp` [1155]
- `F_VN_SobelFilterExp` [1159]
- `F_VN_WarpAffineExp` [1020]
- `F_VN_WarpPerspectiveExp` [1025]

### 6.2.4.7 ETcVnCameraState

Specifies the state of a camera controller, which controls an attached camera.

### Syntax

**Definition:**

```plaintext
TYPE ETcVnCameraState :
{
    TCVN_CS_ERROR := -1,
    TCVN_CS_INITIAL := 0,
    TCVN_CS_INITIALIZING := 1,
    TCVN_CS_INITIALIZED := 2,
    TCVN_CS_OPENING := 3,
    TCVN_CS_OPENED := 4,
    TCVN_CS_STARTACQUISITION := 5,
    TCVN_CS_ACQUIRING := 6,
    TCVN_CS_STOPACQUISITION := 7,
    TCVN_CS_RESETTINGFEATURES := 8,
    TCVN_CS_TRIGGERING := 9,
    TCVN_CS_CLOSING := 10
) DINT;
END_TYPE
```
### Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_CS_ERROR</td>
<td>The camera controller is in an error state (e.g. the connection to the camera was lost).</td>
</tr>
<tr>
<td>TCVN_CSInicial</td>
<td>The camera controller is in the initial state and ready to establish the connection to the camera.</td>
</tr>
<tr>
<td>TCVN_CS_INITIALIZING</td>
<td>The camera controller is initializing the camera.</td>
</tr>
<tr>
<td>TCVN_CS_INITIALIZED</td>
<td>The camera controller has initialized the camera.</td>
</tr>
<tr>
<td>TCVN_CS_OPENING</td>
<td>The camera controller is establishing the connection to the camera.</td>
</tr>
<tr>
<td>TCVN_CS_OPENED</td>
<td>The connection to the camera has been established and the camera controller is ready to start the image acquisition.</td>
</tr>
<tr>
<td>TCVN_CS_STARTACQUISITION</td>
<td>The camera controller is starting the image acquisition.</td>
</tr>
<tr>
<td>TCVN_CS_ACQUIRING</td>
<td>The camera is sending images (either streaming or manually triggered, depending on the configuration).</td>
</tr>
<tr>
<td>TCVN_CS_STOPACQUISITION</td>
<td>The camera controller is stopping the image acquisition.</td>
</tr>
<tr>
<td>TCVN_CS_RESETTINGFEATURES</td>
<td>The camera controller is resetting the features on the camera.</td>
</tr>
<tr>
<td>TCVN_CS_TRIGGERING</td>
<td>The camera controller is processing a software trigger.</td>
</tr>
<tr>
<td>TCVN_CS_CLOSING</td>
<td>The camera controller is closing the connection to the camera.</td>
</tr>
</tbody>
</table>

### Further information

The state machines used for these camera states are described in the respective function blocks for image acquisition:

- FB VN GevCameraControl
- FB VN FileSourceControl
- FB VN SimpleCameraControl

### 6.2.4.8 ETcVnClusteringAlgorithm

Offers clustering algorithms

#### Syntax

**Definition:**

```plaintext
definition ETcVnClusteringAlgorithm : ( 
  TCVN_CA_KMEANSPP := 0, 
  TCVN_CA_LBG := 1 
) DINT;
end_definition
```

#### Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_CA_KMEANSPP</td>
<td>KMeans++ (fixed number of clusters)</td>
</tr>
<tr>
<td>TCVN_CA_LBG</td>
<td>LBG variant (dynamic number of clusters)</td>
</tr>
</tbody>
</table>
6.2.4.9 ETcVnColorMap

Offers color maps (similar to GNU Octave/MATLAB types).

Syntax

Definition:

```plaintext
TYPE ETcVnColorMap :
 {
   TCVN_CM_AUTUMN  :=  0,
   TCVN_CM_BONE    :=  1,
   TCVN_CM_JET     :=  2,
   TCVN_CM_WINTER  :=  3,
   TCVN_CM_RAINBOW :=  4,
   TCVN_CM_OCEAN   :=  5,
   TCVN_CM_SUMMER  :=  6,
   TCVN_CM_SPRING  :=  7,
   TCVN_CM_COOL    :=  8,
   TCVN_CM_HSV     :=  9,
   TCVN_CM_HOT     := 11
 } DINT;
END_TYPE
```

Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_CM_AUTUMN</td>
<td>Red - orange - yellow</td>
</tr>
<tr>
<td>TCVN_CM_BONE</td>
<td>Black - blueish gray - white</td>
</tr>
<tr>
<td>TCVN_CM_JET</td>
<td>Dark blue - green - dark red</td>
</tr>
<tr>
<td>TCVN_CM_WINTER</td>
<td>Blue - green</td>
</tr>
<tr>
<td>TCVN_CM_RAINBOW</td>
<td>Red - green - purple</td>
</tr>
<tr>
<td>TCVN_CM_OCEAN</td>
<td>Black - blue - white</td>
</tr>
<tr>
<td>TCVN_CM_SUMMER</td>
<td>Green - yellow</td>
</tr>
<tr>
<td>TCVN_CM_SPRING</td>
<td>Pink - yellow</td>
</tr>
<tr>
<td>TCVN_CM_COOL</td>
<td>Cyan - magenta</td>
</tr>
<tr>
<td>TCVN_CM_HSV</td>
<td>Red - green - blue - red</td>
</tr>
<tr>
<td>TCVN_CM_HOT</td>
<td>Black - red - yellow - white</td>
</tr>
</tbody>
</table>

Color gradients

The enum ETcVnColorMap defines color gradients for the creation of color tables with the help of the function F_VN_GenerateColorMap [§ 1084]. The following color gradients are predefined:
<table>
<thead>
<tr>
<th>NONE</th>
<th><img src="image" alt="Gray Scale" /></th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTUMN</td>
<td><img src="image" alt="Autumn Gradient" /></td>
</tr>
<tr>
<td>BONE</td>
<td><img src="image" alt="Bone Gradient" /></td>
</tr>
<tr>
<td>JET</td>
<td><img src="image" alt="Jet Gradient" /></td>
</tr>
<tr>
<td>WINTER</td>
<td><img src="image" alt="Winter Gradient" /></td>
</tr>
<tr>
<td>RAINBOW</td>
<td><img src="image" alt="Rainbow Gradient" /></td>
</tr>
<tr>
<td>OCEAN</td>
<td><img src="image" alt="Ocean Gradient" /></td>
</tr>
</tbody>
</table>
Related functions

- F_VN.GenerateColorMap [1084]
- F_VN.GenerateCustomColorMap [1086]

6.2.4.10  ETcVnColorMapSize

Offers color map sizes.

Syntax

Definition:

```plaintext
TYPE ETcVnColorMapSize :
{
    TCVN_CMS_256 := 256,
    TCVN_CMS_65536 := 65536
} UDINT;
END_TYPE
```
6.2.4.11  ETcVnColorSpaceTransform

Offers color space transformations.

Syntax

Definition:

```c
TYPE ETcVnColorSpaceTransform : 
{
    TCVN_CST_BGR_TO_BGRA := 0,
    TCVN_CST_RGB_TO_RGBA := 0,
    TCVN_CST_BGRA_TO_BGR := 1,
    TCVN_CST_RGBA_TO_RGB := 1,
    TCVN_CST_BGR_TO_RGBA := 2,
    TCVN_CST_RGB_TO_BGRA := 2,
    TCVN_CST_BGRA_TO_RGB := 3,
    TCVN_CST_RGBA_TO_BGR := 3,
    TCVN_CST_BGR_TO_RGB := 4,
    TCVN_CST_RGB_TO_BGR := 4,
    TCVN_CST_BGRA_TO_RGBA := 5,
    TCVN_CST_RGBA_TO_BGRA := 5,
    TCVN_CST_BGR_TO_GRAY := 6,
    TCVN_CST_RGB_TO_GRAY := 7,
    TCVN_CST_GRAY_TO_BGR := 8,
    TCVN_CST_GRAY_TO_RGB := 8,
    TCVN_CST_GRAY_TO_BGRA := 9,
    TCVN_CST_GRAY_TO_RGBA := 9,
    TCVN_CST_BGRA_TO_GRAY := 10,
    TCVN_CST_RGBA_TO_GRAY := 11,
    TCVN_CST_BGR_TO_BGR_565 := 12,
    TCVN_CST_BGR_TO_BGR_565 := 13,
    TCVN_CST_BGR_565_TO_RGB := 14,
    TCVN_CST_BGR_565_TO_BGR := 15,
    TCVN_CST_BGR_565_TO_RGBA := 16,
    TCVN_CST_BGR_565_TO_BGRA := 17,
    TCVN_CST_BGR_555_TO_RGBA := 18,
    TCVN_CST_BGR_555_TO_BGRA := 19,
    TCVN_CST_BGR_555_TO_GRAY := 20,
    TCVN_CST_BGR_555_TO_GRAY := 21,
    TCVN_CST_BGR_555_TO_BGR := 22,
    TCVN_CST_BGR_555_TO_BGR := 23,
    TCVN_CST_BGR_555_TO_GRAY := 24,
    TCVN_CST_BGR_555_TO_BGR := 25,
    TCVN_CST_BGR_555_TO_BGR := 26,
    TCVN_CST_BGR_555_TO_BGR := 27,
    TCVN_CST_BGR_555_TO_RGBA := 28,
    TCVN_CST_BGR_555_TO_BGRA := 29,
    TCVN_CST_BGR_555_TO_GRAY := 30,
    TCVN_CST_BGR_555_TO_GRAY := 31,
    TCVN_CST_BGR_TO_XYZ := 32,
    TCVN_CST_BGR_TO_XYZ := 33,
    TCVN_CST_BGR_TO_XYZ := 34,
    TCVN_CST_BGR_TO_XYZ := 35,
    TCVN_CST_BGR_TO_XYZ := 36,
    TCVN_CST_BGR_TO_XYZ := 37,
    TCVN_CST_BGR_TO_XYZ := 38,
    TCVN_CST_BGR_TO_XYZ := 39,
    TCVN_CST_BGR_TO_HSV := 40,
    TCVN_CST_BGR_TO_HSV := 41,
}```
<table>
<thead>
<tr>
<th>Function</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_CST_BGR_TO_LAB</td>
<td>44</td>
</tr>
<tr>
<td>TCVN_CST_RGB_TO_LAB</td>
<td>45</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_RG_TO_BGR</td>
<td>46</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_BG_TO_BGR</td>
<td>46</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_GR_TO_BGR</td>
<td>47</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_GB_TO_BGR</td>
<td>47</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_BG_TO_BGR</td>
<td>49</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_GB_TO_BGR</td>
<td>49</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_GR_TO_BGR</td>
<td>49</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_BG_TO_BGR</td>
<td>50</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_GB_TO_BGR</td>
<td>51</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_BG_TO_BGR</td>
<td>52</td>
</tr>
<tr>
<td>TCVN_CST_RGB_TO_LHS</td>
<td>53</td>
</tr>
<tr>
<td>TCVN_CST_HSV_TO_BGR</td>
<td>54</td>
</tr>
<tr>
<td>TCVN_CST_HSV_TO_RGB</td>
<td>55</td>
</tr>
<tr>
<td>TCVN_CST_LAB_TO_BGR</td>
<td>56</td>
</tr>
<tr>
<td>TCVN_CST_LAB_TO_RGB</td>
<td>57</td>
</tr>
<tr>
<td>TCVN_CST_LUV_TO_BGR</td>
<td>58</td>
</tr>
<tr>
<td>TCVN_CST_LUV_TO_RGB</td>
<td>59</td>
</tr>
<tr>
<td>TCVN_CST_HLS_TO_BGR</td>
<td>60</td>
</tr>
<tr>
<td>TCVN_CST_HLS_TO_RGB</td>
<td>61</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_RG_TO_BGR_VNG</td>
<td>62</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_GR_TO_BGR_VNG</td>
<td>63</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_BG_TO_BGR_VNG</td>
<td>64</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_GB_TO_BGR_VNG</td>
<td>65</td>
</tr>
<tr>
<td>TCVN_CST_BGR_TO_HSV_FULL</td>
<td>66</td>
</tr>
<tr>
<td>TCVN_CST_RGB_TO_HSV_FULL</td>
<td>67</td>
</tr>
<tr>
<td>TCVN_CST_BGR_TO_HLS_FULL</td>
<td>68</td>
</tr>
<tr>
<td>TCVN_CST_RGB_TO_HLS_FULL</td>
<td>69</td>
</tr>
<tr>
<td>TCVN_CST_HSV_TO_BGR_FULL</td>
<td>70</td>
</tr>
<tr>
<td>TCVN_CST_HSV_TO_RGB_FULL</td>
<td>71</td>
</tr>
<tr>
<td>TCVN_CST_HLS_TO_BGR_FULL</td>
<td>72</td>
</tr>
<tr>
<td>TCVN_CST_HLS_TO_RGB_FULL</td>
<td>73</td>
</tr>
<tr>
<td>TCVN_CST_LBGR_TO_LAB</td>
<td>74</td>
</tr>
<tr>
<td>TCVN_CST_LRGB_TO_LAB</td>
<td>75</td>
</tr>
<tr>
<td>TCVN_CST_LBGR_TO_LUV</td>
<td>76</td>
</tr>
<tr>
<td>TCVN_CST_LRGB_TO_LUV</td>
<td>77</td>
</tr>
<tr>
<td>TCVN_CST_LAB_TO_LBGR</td>
<td>78</td>
</tr>
<tr>
<td>TCVN_CST_LAB_TO_LRGB</td>
<td>79</td>
</tr>
<tr>
<td>TCVN_CST_LUV_TO_LBGR</td>
<td>80</td>
</tr>
<tr>
<td>TCVN_CST_LUV_TO_LRGB</td>
<td>81</td>
</tr>
<tr>
<td>TCVN_CST_BGR_TO_YUV</td>
<td>82</td>
</tr>
<tr>
<td>TCVN_CST_BGR_TO_YUV</td>
<td>83</td>
</tr>
<tr>
<td>TCVN_CST_RGB_TO_YUV</td>
<td>84</td>
</tr>
<tr>
<td>TCVN_CST_RGB_TO_YUV</td>
<td>85</td>
</tr>
<tr>
<td>TCVN_CST_BGR_TO_GRAY</td>
<td>86</td>
</tr>
<tr>
<td>TCVN_CST_BGR_TO_GRAY</td>
<td>87</td>
</tr>
<tr>
<td>TCVN_CST_RGB_TO_GRAY</td>
<td>88</td>
</tr>
<tr>
<td>TCVN_CST_BGR_TO_GRAY</td>
<td>89</td>
</tr>
<tr>
<td>TCVN_CST_RGB_TO_GRAY</td>
<td>90</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_NV12_TO_RGB</td>
<td>91</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_NV21_TO_RGB</td>
<td>92</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_NV21_TO_BGR</td>
<td>93</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_SP_TO_RGB</td>
<td>94</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_SP_TO_BGR</td>
<td>95</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_NV12_TO_RGBA</td>
<td>96</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_NV21_TO_RGBA</td>
<td>97</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_NV21_TO_BGRA</td>
<td>98</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_V12_TO_BGR</td>
<td>99</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_1YUV_TO_RGB</td>
<td>100</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_1YUV_TO_BGR</td>
<td>101</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_1420_TO_RGB</td>
<td>102</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_1420_TO_BGR</td>
<td>103</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_1420_TO_BGRA</td>
<td>104</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_1420_TO_BGRA</td>
<td>105</td>
</tr>
</tbody>
</table>
TCVN_CST_YUV_420_P_TO_RGBA := 102,
TCVN_CST_YUV_420_P_TO_BGRA := 103,
TCVN_CST_YUV_420_TO_GRAY := 106,
TCVN_CST_YUV_420_NV21_TO_GRAY := 106,
TCVN_CST_YUV_420_NV12_TO_GRAY := 106,
TCVN_CST_YUV_420_NV12_TO_GRAY := 106,
TCVN_CST_YUV_420_I420_TO_GRAY := 106,
TCVN_CST_YUV_420_SP_TO_GRAY := 106,
TCVN_CST_YUV_420_P_TOGRAY := 106,
TCVN_CST_YUV_422_UYVY_TO_RGB := 107,
TCVN_CST_YUV_422_UYVY_TO_BGR := 108,
TCVN_CST_YUV_422_Y422_TO_RGB := 107,
TCVN_CST_YUV_422_Y422_TO_BGR := 108,
TCVN_CST_YUV_422_UYNV_TO_RGB := 107,
TCVN_CST_YUV_422_UYNV_TO_BGR := 108,
TCVN_CST_YUV_422_UYVY_TO_RGBA := 111,
TCVN_CST_YUV_422_UYVY_TO_BGRA := 112,
TCVN_CST_YUV_422_Y422_TO_RGBA := 111,
TCVN_CST_YUV_422_Y422_TO_BGRA := 112,
TCVN_CST_YUV_422_UYNV_TO_RGBA := 111,
TCVN_CST_YUV_422_UYNV_TO_BGRA := 112,
TCVN_CST_YUV_422_YUY2_TO_RGB := 115,
TCVN_CST_YUV_422_YUY2_TO_BGR := 116,
TCVN_CST_YUV_422_YVYU_TO_RGB := 117,
TCVN_CST_YUV_422_YVYU_TO_BGR := 118,
TCVN_CST_YUV_422_YUYV_TO_RGB := 115,
TCVN_CST_YUV_422_YUYV_TO_BGR := 116,
TCVN_CST_YUV_422_YUV2_TO_BGR := 112,
TCVN_CST_YUV_422_YUV2_TO_BGR := 119,
TCVN_CST_YUV_422_YVYU_TO_GRAY := 123,
TCVN_CST_YUV_422_YUYU_TO_GRAY := 124,
TCVN_CST_YUV_422_Y422_TO_GRAY := 123,
TCVN_CST_YUV_422_UYNV_TO_GRAY := 123,
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_CST_BGR_TO_BGRA</td>
<td>Transform the color space from BGR to BGRA.</td>
</tr>
<tr>
<td>TCVN_CST_RGB_TO_RGBA</td>
<td>Transform the color space from RGB to RGBA.</td>
</tr>
<tr>
<td>TCVN_CST_BGRA_TO_BGR</td>
<td>Transform the color space from BGRA to BGR.</td>
</tr>
<tr>
<td>TCVN_CST_RGBA_TO_RGB</td>
<td>Transform the color space from RGBA to RGB.</td>
</tr>
<tr>
<td>TCVN_CST_BGR_TO_RGBA</td>
<td>Transform the color space from BGR to RGBA.</td>
</tr>
<tr>
<td>TCVN_CST_RGBA_TO_BGR</td>
<td>Transform the color space from RGBA to BGR.</td>
</tr>
<tr>
<td>TCVN_CST_BGRA_TO_RGB</td>
<td>Transform the color space from BGRA to RGB.</td>
</tr>
<tr>
<td>TCVN_CST_BGR_TO_GRAY</td>
<td>Transform the color space from BGR to Gray.</td>
</tr>
<tr>
<td>TCVN_CST_RGB_TO_GRAY</td>
<td>Transform the color space from RGB to Gray.</td>
</tr>
<tr>
<td>TCVN_CST_GRAY_TO_BGR</td>
<td>Transform the color space from Gray to BGR.</td>
</tr>
<tr>
<td>TCVN_CST_GRAY_TO_RGB</td>
<td>Transform the color space from Gray to RGB.</td>
</tr>
<tr>
<td>TCVN_CST_BGRA_TO_BGR</td>
<td>Transform the color space from BGRA to BGR.</td>
</tr>
<tr>
<td>TCVN_CST_BGRA_TO_RGB</td>
<td>Transform the color space from BGRA to RGB.</td>
</tr>
<tr>
<td>TCVN_CST_BGRA_TO_GRAY</td>
<td>Transform the color space from BGRA to Gray.</td>
</tr>
<tr>
<td>TCVN_CST_BGR_TO_RGB_565</td>
<td>Transform the color space from RGB(24 bit 888) to BGR(16 bit 565).</td>
</tr>
<tr>
<td>TCVN_CST_BGR_TO_BGR_565</td>
<td>Transform the color space from BGR(24 bit 888) to BGR(16 bit 565).</td>
</tr>
<tr>
<td>TCVN_CST_BGR_565_TO_RGB</td>
<td>Transform the color space from BGR(16 bit 565) to RGB(24 bit 888).</td>
</tr>
<tr>
<td>TCVN_CST_BGR_565_TO_BGR</td>
<td>Transform the color space from BGR(16 bit 565) to BGR(24 bit 888).</td>
</tr>
<tr>
<td>TCVN_CST_RGBA_TO_BGR_565</td>
<td>Transform the color space from RGBA(32 bit 8888) to BGR(16 bit 565).</td>
</tr>
<tr>
<td>TCVN_CST_BGRA_TO_BGR_565</td>
<td>Transform the color space from BGRA(32 bit 8888) to BGR(16 bit 565).</td>
</tr>
<tr>
<td>TCVN_CST_BGR_565_TO_RGBA</td>
<td>Transform the color space from BGR(16 bit 565) to RGBA(32 bit 8888).</td>
</tr>
<tr>
<td>TCVN_CST_BGR_565_TO_BGRA</td>
<td>Transform the color space from BGR(16 bit 565) to BGRA(32 bit 8888).</td>
</tr>
<tr>
<td>TCVN_CST_GRAY_TO_BGR_565</td>
<td>Transform the color space from Gray to BGR(16 bit 565).</td>
</tr>
<tr>
<td>TCVN_CST_BGR_555_TO_RGB</td>
<td>Transform the color space from BGR(16 bit 555) to RGB(24 bit 888).</td>
</tr>
<tr>
<td>TCVN_CST_BGR_555_TO_BGR</td>
<td>Transform the color space from BGR(16 bit 555) to BGR(24 bit 888).</td>
</tr>
<tr>
<td>TCVN_CST_RGBA_TO_BGR_555</td>
<td>Transform the color space from RGBA(32 bit 8888) to BGR(16 bit 555).</td>
</tr>
<tr>
<td>TCVN_CST_BGRA_TO_BGR_555</td>
<td>Transform the color space from BGRA(32 bit 8888) to BGR(16 bit 555).</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TCVN_CST_BGR_555_TO_RGBA</td>
<td>Transform the color space from BGR(16 bit 555) to RGBA(32 bit 8888).</td>
</tr>
<tr>
<td>TCVN_CST_BGR_555_TO_BGRA</td>
<td>Transform the color space from BGR(16 bit 555) to BGRA(32 bit 8888).</td>
</tr>
<tr>
<td>TCVN_CST_GRAY_TO_BGR_555</td>
<td>Transform the color space from Gray to BGR(16 bit 555).</td>
</tr>
<tr>
<td>TCVN_CST_BGR_555_TO_GRAY</td>
<td>Transform the color space from BGR(16 bit 555) to Gray.</td>
</tr>
<tr>
<td>TCVN_CST_BGR_TO_XYZ</td>
<td>Transform the color space from BGR to CIE XYZ (scaled to the full value range of the image).</td>
</tr>
<tr>
<td>TCVN_CST_RGB_TO_XYZ</td>
<td>Transform the color space from RGB to CIE XYZ (scaled to the full value range of the image).</td>
</tr>
<tr>
<td>TCVN_CST_XYZ_TO_BGR</td>
<td>Transform the color space from CIE XYZ (scaled to the full value range of the image) to BGR.</td>
</tr>
<tr>
<td>TCVN_CST_XYZ_TO_RGB</td>
<td>Transform the color space from CIE XYZ (scaled to the full value range of the image) to RGB.</td>
</tr>
<tr>
<td>TCVN_CST_BGR_TO_YCRCB</td>
<td>Transform the color space from BGR to YCrCb (scaled to the full value range of the image).</td>
</tr>
<tr>
<td>TCVN_CST_RGB_TO_YCRCB</td>
<td>Transform the color space from RGB to YCrCb (scaled to the full value range of the image).</td>
</tr>
<tr>
<td>TCVN_CST_YCRCB_TO_BGR</td>
<td>Transform the color space from YCrCb (scaled to the full value range of the image) to BGR.</td>
</tr>
<tr>
<td>TCVN_CST_YCRCB_TO_RGB</td>
<td>Transform the color space from YCrCb (scaled to the full value range of the image) to RGB.</td>
</tr>
<tr>
<td>TCVN_CST_BGR_TO_HSV</td>
<td>Transform the color space from BGR to HSV (for images of type USINT, H is scaled to a range from 0 to 180 and S,V from 0 to 255).</td>
</tr>
<tr>
<td>TCVN_CST_RGB_TO_HSV</td>
<td>Transform the color space from RGB to HSV (for images of type USINT, H is scaled to a range from 0 to 180 and S,V from 0 to 255).</td>
</tr>
<tr>
<td>TCVN_CST_BGR_TO_LAB</td>
<td>Transform the color space from BGR to CIE L<em>a</em>b* (for images of type USINT, all channels are scaled to a range from 0 to 255).</td>
</tr>
<tr>
<td>TCVN_CST_RGB_TO_LAB</td>
<td>Transform the color space from RGB to CIE L<em>a</em>b* (for images of type USINT, all channels are scaled to a range from 0 to 255).</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_RG_TO_BGR</td>
<td>Transform the color space from BayerRG to BGR.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_BG_TO_RGB</td>
<td>Transform the color space from BayerBG to RGB.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_GR_TO_BGR</td>
<td>Transform the color space from BayerGR to BGR.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_BG_TO_RGB</td>
<td>Transform the color space from BayerBG to RGB.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_BG_TO_RGB</td>
<td>Transform the color space from BayerBG to RGB.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_GR_TO_RGB</td>
<td>Transform the color space from BayerGR to RGB.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_GB_TO_BGR</td>
<td>Transform the color space from BayerGB to BGR.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_GB_TO_RGB</td>
<td>Transform the color space from BayerGB to RGB.</td>
</tr>
<tr>
<td>TCVN_CST_BGR_TO_LUV</td>
<td>Transform the color space from BGR to CIE L<em>u</em>v* (for images of type USINT, all channels are scaled to a range from 0 to 255).</td>
</tr>
<tr>
<td>TCVN_CST_RGB_TO_LUV</td>
<td>Transform the color space from RGB to CIE L<em>u</em>v* (for images of type USINT, all channels are scaled to a range from 0 to 255).</td>
</tr>
<tr>
<td>TCVN_CST_BGR_TO_HLS</td>
<td>Transform the color space from BGR to HLS (for images of type USINT, H is scaled to a range from 0 to 180 and L,S from 0 to 255).</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TCVN_CST_RGB_TO_HLS</td>
<td>Transform the color space from RGB to HLS (for images of type USINT, H is scaled to a range from 0 to 180 and L,S from 0 to 255).</td>
</tr>
<tr>
<td>TCVN_CST_HSV_TO_BGR</td>
<td>Transform the color space from HSV (for images of type USINT, H is scaled to a range from 0 to 180 and S,V from 0 to 255) to BGR.</td>
</tr>
<tr>
<td>TCVN_CST_HSV_TO_RGB</td>
<td>Transform the color space from HSV (for images of type USINT, H is scaled to a range from 0 to 180 and S,V from 0 to 255) to RGB.</td>
</tr>
<tr>
<td>TCVN_CST_LAB_TO_BGR</td>
<td>Transform the color space from CIE L<em>a</em>b* (for images of type USINT, all channels are scaled to a range from 0 to 255) to BGR.</td>
</tr>
<tr>
<td>TCVN_CST_LAB_TO_RGB</td>
<td>Transform the color space from CIE L<em>a</em>b* (for images of type USINT, all channels are scaled to a range from 0 to 255) to RGB.</td>
</tr>
<tr>
<td>TCVN_CST_LUV_TO_BGR</td>
<td>Transform the color space from CIE L<em>u</em>v* (for images of type USINT, all channels are scaled to a range from 0 to 255) to BGR.</td>
</tr>
<tr>
<td>TCVN_CST_LUV_TO_RGB</td>
<td>Transform the color space from CIE L<em>u</em>v* (for images of type USINT, all channels are scaled to a range from 0 to 255) to RGB.</td>
</tr>
<tr>
<td>TCVN_CST_HLS_TO_BGR</td>
<td>Transform the color space from HLS (for images of type USINT, H is scaled to a range from 0 to 180 and L,S from 0 to 255) to BGR.</td>
</tr>
<tr>
<td>TCVN_CST_HLS_TO_RGB</td>
<td>Transform the color space from HLS (for images of type USINT, H is scaled to a range from 0 to 180 and L,S from 0 to 255) to RGB.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_RG_TO_BGR_VNG</td>
<td>Transform the color space from BayerRG to BGR, using Variable Number of Gradients.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_GR_TO_BGR_VNG</td>
<td>Transform the color space from BayerGR to BGR, using Variable Number of Gradients.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_BG_TO_BGR_VNG</td>
<td>Transform the color space from BayerBG to BGR, using Variable Number of Gradients.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_GB_TO_BGR_VNG</td>
<td>Transform the color space from BayerGB to BGR, using Variable Number of Gradients.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_RG_TO_RGB_VNG</td>
<td>Transform the color space from BayerRG to RGB, using Variable Number of Gradients.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_GR_TO_RGB_VNG</td>
<td>Transform the color space from BayerGR to RGB, using Variable Number of Gradients.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_BG_TO_RGB_VNG</td>
<td>Transform the color space from BayerBG to RGB, using Variable Number of Gradients.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_GB_TO_RGB_VNG</td>
<td>Transform the color space from BayerGB to RGB, using Variable Number of Gradients.</td>
</tr>
<tr>
<td>TCVN_CST_BGR_TO_HSV_FULL</td>
<td>Transform the color space from BGR to HSV (for images of type USINT, all channels are scaled to a range from 0 to 255).</td>
</tr>
<tr>
<td>TCVN_CST_RGB_TO_HSV_FULL</td>
<td>Transform the color space from RGB to HSV (for images of type USINT, all channels are scaled to a range from 0 to 255).</td>
</tr>
<tr>
<td>TCVN_CST_BGR_TO_HLS_FULL</td>
<td>Transform the color space from BGR to HLS (for images of type USINT, all channels are scaled to a range from 0 to 255).</td>
</tr>
<tr>
<td>TCVN_CST_RGB_TO_HLS_FULL</td>
<td>Transform the color space from RGB to HLS (for images of type USINT, all channels are scaled to a range from 0 to 255).</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TCVN_CST_HSV_TO_BGR_FULL</td>
<td>Transform the color space from HSV (for images of type USINT, all channels are scaled to a range from 0 to 255) to BGR.</td>
</tr>
<tr>
<td>TCVN_CST_HSV_TO_RGB_FULL</td>
<td>Transform the color space from HSV (for images of type USINT, all channels are scaled to a range from 0 to 255) to RGB.</td>
</tr>
<tr>
<td>TCVN_CST_HLS_TO_BGR_FULL</td>
<td>Transform the color space from HLS (for images of type USINT, all channels are scaled to a range from 0 to 255) to BGR.</td>
</tr>
<tr>
<td>TCVN_CST_HLS_TO_RGB_FULL</td>
<td>Transform the color space from HLS (for images of type USINT, all channels are scaled to a range from 0 to 255) to RGB.</td>
</tr>
<tr>
<td>TCVN_CST_LBGR_TO_LAB</td>
<td>Transform the color space from LBGR to CIE L<em>a</em>b* (for images of type USINT, all channels are scaled to a range from 0 to 255).</td>
</tr>
<tr>
<td>TCVN_CST_LRGB_TO_LAB</td>
<td>Transform the color space from LRGB to CIE L<em>a</em>b* (for images of type USINT, all channels are scaled to a range from 0 to 255).</td>
</tr>
<tr>
<td>TCVN_CST_LBGR_TO_LUV</td>
<td>Transform the color space from LBGR to CIE L<em>u</em>v* (for images of type USINT, all channels are scaled to a range from 0 to 255).</td>
</tr>
<tr>
<td>TCVN_CST_LRGB_TO_LUV</td>
<td>Transform the color space from LRGB to CIE L<em>u</em>v* (for images of type USINT, all channels are scaled to a range from 0 to 255).</td>
</tr>
<tr>
<td>TCVN_CST_LAB_TO_LBGR</td>
<td>Transform the color space from CIE L<em>a</em>b* (for images of type USINT, all channels are scaled to a range from 0 to 255) to LBGR.</td>
</tr>
<tr>
<td>TCVN_CST_LAB_TO_LRGB</td>
<td>Transform the color space from CIE L<em>a</em>b* (for images of type USINT, all channels are scaled to a range from 0 to 255) to LRGB.</td>
</tr>
<tr>
<td>TCVN_CST_LUV_TO_LBGR</td>
<td>Transform the color space from CIE L<em>u</em>v* (for images of type USINT, all channels are scaled to a range from 0 to 255) to LBGR.</td>
</tr>
<tr>
<td>TCVN_CST_LUV_TO_LRGB</td>
<td>Transform the color space from CIE L<em>u</em>v* (for images of type USINT, all channels are scaled to a range from 0 to 255) to LRGB.</td>
</tr>
<tr>
<td>TCVN_CST_BGR_TO_YUV</td>
<td>Transform the color space from BGR to YUV (scaled to the full value range of the image).</td>
</tr>
<tr>
<td>TCVN_CST_RGB_TO_YUV</td>
<td>Transform the color space from RGB to YUV (scaled to the full value range of the image).</td>
</tr>
<tr>
<td>TCVN_CST_YUV_TO_BGR</td>
<td>Transform the color space from YUV (scaled to the full value range of the image) to BGR.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_TO_RGB</td>
<td>Transform the color space from YUV (scaled to the full value range of the image) to RGB.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_RG_TO_GRAY</td>
<td>Transform the color space from BayerRG to Gray.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_GR_TO_GRAY</td>
<td>Transform the color space from BayerGR to Gray.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_BG_TO_GRAY</td>
<td>Transform the color space from BayerBG to Gray.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_GB_TO_GRAY</td>
<td>Transform the color space from BayerGB to Gray.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_NV12_TO_RGB</td>
<td>Transform the color space from YUV420 NV12 to RGB.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_NV12_TO_BGR</td>
<td>Transform the color space from YUV420 NV12 to BGR.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_NV21_TO_RGB</td>
<td>Transform the color space from YUV420 NV21 (SP) to RGB.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_NV21_TO_BGR</td>
<td>Transform the color space from YUV420 NV21 (SP) to BGR.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_SP_TO_RGB</td>
<td>Transform the color space from YUV420 NV21 (SP) to RGB.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_SP_TO_BGR</td>
<td>Transform the color space from YUV420 NV21 (SP) to RGB.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_NV12_TO_RGBA</td>
<td>Transform the color space from YUV420 NV12 to RGBA.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_NV12_TO_BGRA</td>
<td>Transform the color space from YUV420 NV12 to BGRA.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_NV21_TO_RGBA</td>
<td>Transform the color space from YUV420 NV21 (SP) to RGBA.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_NV21_TO_BGRA</td>
<td>Transform the color space from YUV420 NV21 (SP) to BGRA.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_SP_TO_RGBA</td>
<td>Transform the color space from YUV420 NV21 (SP) to RGBA.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_SP_TO_BGRA</td>
<td>Transform the color space from YUV420 NV21 (SP) to BGRA.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_YUV12_TO_RGB</td>
<td>Transform the color space from YUV420 YY12 (P) to RGB.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_YUV12_TO_BGR</td>
<td>Transform the color space from YUV420 YY12 (P) to BGR.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_IYUV_TO_RGB</td>
<td>Transform the color space from YUV420 IYUV (I420) to RGB.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_IYUV_TO_BGR</td>
<td>Transform the color space from YUV420 IYUV (I420) to BGR.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_I420_TO_RGB</td>
<td>Transform the color space from YUV420 IYUV (I420) to RGB.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_I420_TO_BGR</td>
<td>Transform the color space from YUV420 IYUV (I420) to BGR.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_P_TO_RGB</td>
<td>Transform the color space from YUV420 YV12 (P) to RGB.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_P_TO_BGR</td>
<td>Transform the color space from YUV420 YV12 (P) to BGR.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_YV12_TO_RGBA</td>
<td>Transform the color space from YUV420 YV12 (P) to RGBA.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_YV12_TO_BGRA</td>
<td>Transform the color space from YUV420 YV12 (P) to BGRA.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_YV12_TO_GRAY</td>
<td>Transform the color space from YUV420 YV12 to Gray.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_NV12_TO_GRAY</td>
<td>Transform the color space from YUV420 NV12 to Gray.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_NV21_TO_GRAY</td>
<td>Transform the color space from YUV420 NV21 to Gray.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_P_TO_GRAY</td>
<td>Transform the color space from YUV420 YV12 (P) to Gray.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_IYUV_TO_GRAY</td>
<td>Transform the color space from YUV420 to Gray.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_I420_TO_GRAY</td>
<td>Transform the color space from YUV420 to Gray.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_SP_TO_GRAY</td>
<td>Transform the color space from YUV420 to Gray.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_420_P_TO_GRAY</td>
<td>Transform the color space from YUV420 to Gray.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_422_UYVV_TO_RGB</td>
<td>Transform the color space from YUV422 UYVY (Y422, UYNV) to RGB.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_422_UYYV_TO_BGR</td>
<td>Transform the color space from YUV422 UYVY (Y422, UYNV) to BGR.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_422_Y422_TO_RGB</td>
<td>Transform the color space from YUV422 UYVY (Y422, UYNV) to RGB.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_422_Y422_TO_BGR</td>
<td>Transform the color space from YUV422 UYVY (Y422, UYNV) to BGR.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_422_Y422_TO_RGBA</td>
<td>Transform the color space from YUV422 UYVY (Y422, UYNV) to RGBA.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_422_Y422_TO_BGRA</td>
<td>Transform the color space from YUV422 UYVY (Y422, UYNV) to BGRA.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_422_YUY2_TO_RGB</td>
<td>Transform the color space from YUV422 YUYV (YUY2, YUNV) to RGB.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_422_YUY2_TO_BGR</td>
<td>Transform the color space from YUV422 YUYV (YUY2, YUNV) to BGR.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_422_YUY2_TO_RGBA</td>
<td>Transform the color space from YUV422 YUYV (YUY2, YUNV) to RGBA.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_422_YUY2_TO_BGRA</td>
<td>Transform the color space from YUV422 YUYV (YUY2, YUNV) to BGRA.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TCVN_CST_YUV_422_YUYV_TO_BGRA</td>
<td>Transform the color from YUV422 YUYV (YUY2, YUNV) to BGRA.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_422_YUNV_TO_RGBA</td>
<td>Transform the color from YUV422 YUYV (YUY2, YUNV) to RGBA.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_422_YUNV_TO_BGRA</td>
<td>Transform the color from YUV422 YUYV (YUY2, YUNV) to BGRA.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_422_UYVV_TO_GRAY</td>
<td>Transform the color from YUV422 UYVV (Y422, UYNV) to Gray.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_422_UYY2_TO_GRAY</td>
<td>Transform the color from YUV422 UYVV (YUY2, YUNV) to Gray.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_422_Y422_TO_GRAY</td>
<td>Transform the color from YUV422 UYVV (Y422, UYNV) to Gray.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_422_UYNV_TO_GRAY</td>
<td>Transform the color from YUV422 UYVV (Y422, UYNV) to Gray.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_422_YVYU_TO_GRAY</td>
<td>Transform the color from YUV422 YVYU to Gray.</td>
</tr>
<tr>
<td>TCVN_CST_YUV_422_YUYV_TO_GRAY</td>
<td>Transform the color from YUV422 YUYV (YUY2, YUNV) to Gray.</td>
</tr>
<tr>
<td>TCVN_CST_CST_RGBA_TO_PREMULTIPLICATED_RGBA</td>
<td>Transform the color space from RGBA to premultiplicated RGBA.</td>
</tr>
<tr>
<td>TCVN_CST_RGBA_TO_YUV_420_I420</td>
<td>Transform the color space from RGBA to YUV420 IYUV (I420).</td>
</tr>
<tr>
<td>TCVN_CST_BGR_TO_YUV_420_I420</td>
<td>Transform the color space from BGR to YUV420 IYUV (I420).</td>
</tr>
<tr>
<td>TCVN_CST_RGBA_TO_YUV_420_IYUV</td>
<td>Transform the color space from RGBA to YUV420 IYUV (I420).</td>
</tr>
<tr>
<td>TCVN_CST_BGR_TO_YUV_420_IYUV</td>
<td>Transform the color space from BGR to YUV420 IYUV (I420).</td>
</tr>
<tr>
<td>TCVN_CST_RGBA_TO_YUV_420_I420</td>
<td>Transform the color space from RGBA to YUV420 IYUV (I420).</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_RG_BAYER_GR_TO_BGR_EA</td>
<td>Transform the color space from BayerRG to BGR, using an Edge Aware algorithm.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_BG_TO_BGR_EA</td>
<td>Transform the color space from BayerBG to BGR, using an Edge Aware algorithm.</td>
</tr>
</tbody>
</table>
### API reference

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_CST_BAYER_GB_TO_BGR_EA</td>
<td>Transform the color space from BayerGB to BGR, using an Edge Aware algorithm.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_RG_TO_RGB_EA</td>
<td>Transform the color space from BayerRG to RGB, using an Edge Aware algorithm.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_GR_TO_RGB_EA</td>
<td>Transform the color space from BayerGR to RGB, using an Edge Aware algorithm.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_BG_TO_RGB_EA</td>
<td>Transform the color space from BayerBG to RGB, using an Edge Aware algorithm.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_GB_TO_RGB_EA</td>
<td>Transform the color space from BayerGB to RGB, using an Edge Aware algorithm.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_RG_TO_BGRA</td>
<td>Transform the color space from BayerRG to BGRA.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_GR_TO_BGRA</td>
<td>Transform the color space from BayerGR to BGRA.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_BG_TO_BGRA</td>
<td>Transform the color space from BayerBG to BGRA.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_GB_TO_BGRA</td>
<td>Transform the color space from BayerGB to BGRA.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_RG_TO_RGBA</td>
<td>Transform the color space from BayerRG to RGBA.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_GR_TO_RGBA</td>
<td>Transform the color space from BayerGR to RGBA.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_BG_TO_RGBA</td>
<td>Transform the color space from BayerBG to RGBA.</td>
</tr>
<tr>
<td>TCVN_CST_BAYER_GB_TO_RGBA</td>
<td>Transform the color space from BayerGB to RGBA.</td>
</tr>
<tr>
<td>TCVN_CST_MAX</td>
<td>For internal use only, adapted when adding new values.</td>
</tr>
</tbody>
</table>

**Related functions**

- F_VN_ConvertColorSpace [p. 1083]

#### 6.2.4.12 ETCVnColorTrainingMethod

Offers color training methods.

**Syntax**

Definition:

```cpp
define TYPE ETCVnColorTrainingMethod :
{
    TCVN_CTM_LAB := 0,
    TCVN_CTM_RGB := 1
}DINT;
END_TYPE
```

**Values**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_CTM_LAB</td>
<td>Trains the color in CIE L<em>a</em>b* color space.</td>
</tr>
<tr>
<td>TCVN_CTM_RGB</td>
<td>Trains the color in RGB color space.</td>
</tr>
</tbody>
</table>

**Related functions**

- F_VN_TrainImageColorExp [p. 1105]

#### 6.2.4.13 ETCVnConnectedComponentsAlgorithm

Offers connected components algorithms.
Syntax

Definition:

```plaintext
TYPE ETcVnConnectedComponentsAlgorithm :
{
     TCVN_CCA_WU := 0,
     TCVN_CCA_GRANA := 1
} DINT;
END_TYPE
```

Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_CCA_WU</td>
<td>SAUF algorithm</td>
</tr>
<tr>
<td>TCVN_CCA_GRANA</td>
<td>BBDT algorithm for 8-way connectivity, SAUF algorithm for 4-way connectivity</td>
</tr>
</tbody>
</table>

6.2.4.14 ETcVnContainerExportFormat

Offers container export formats.

Syntax

Definition:

```plaintext
TYPE ETcVnContainerExportFormat :
{
     TCVN_CEF_XML := 0,
     TCVN_CEF_XML_SERIALIZED := 1,
     TCVN_CEF_CSV := 2
} UINT;
END_TYPE
```

Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_CEF_XML</td>
<td>XML format with human readable data</td>
</tr>
<tr>
<td>TCVN_CEF_XML_SERIALIZED</td>
<td>XML format with serialized data</td>
</tr>
<tr>
<td>TCVN_CEF_CSV</td>
<td>CSV format (limited to two dimensional representation, so not applicable for all container types)</td>
</tr>
</tbody>
</table>

Related elements

- FB_VN_WriteContainer [1308]

6.2.4.15 ETcVnContourApproximationMethod

Offers methods for contour approximation.

Syntax

Definition:

```plaintext
TYPE ETcVnContourApproximationMethod :
{
     TCVN_CAM_NONE := 1,
     TCVN_CAM_SIMPLE := 2,
     TCVN_CAM_TC89_L1 := 3,
     TCVN_CAM_TC89_KCOS := 4
} DINT;
END_TYPE
```
# Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_CAM_NONE</td>
<td>No approximation, every single point is stored, i.e. two subsequent points are always direct neighbors (horizontal, vertical or diagonal).</td>
</tr>
<tr>
<td>TCVN_CAM_SIMPLE</td>
<td>Lossless compression of segments that form straight lines in horizontal, vertical or diagonal direction leaving only their endpoints (For instance, an up-right rectangle is reduced to its four corner points.)</td>
</tr>
</tbody>
</table>

**Further information**

This type indicates whether and if so how the points describing a contour are to be simplified.

- **TCVN_CAM_NONE**
  
  No simplification of the contour description, the contours are described by all enclosing points.
• **TCVN_CAM_SIMPLE**
  Lossless simplification

• **TCVN_CAM_TC89_L1**
• **TCVN_CAM_TC89_KCOS**


![Image of contour simplification](image.png)

**Related functions**

- `F_VN_FindContoursExp` [1040]
- `F_VN_FindContourHierarchyExp` [1033]

### 6.2.4.16 **ETcVnContourRetrievalMode**

Offers retrieval modes for a contour search.

**Syntax**

**Definition:**

```plaintext
TYPE ETcVnContourRetrievalMode :
{
  TCVN_CRM_EXTERNAL := 0,
  TCVN_CRM_LIST := 1,
  TCVN_CRM_CONNECTED_COMPONENTS := 2,
  TCVN_CRM_TREE := 3,
  TCVN_CRM_FLOODFILL := 4
} DINT;
END_TYPE
```

**Values**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_CRM_EXTERNAL</td>
<td>Returns only external contours.</td>
</tr>
<tr>
<td>TCVN_CRM_LIST</td>
<td>Returns all found contours, not considering their hierarchy.</td>
</tr>
<tr>
<td>TCVN_CRM_CONNECTED_COMPONENTS</td>
<td>Returns all contours with a 2-level hierarchy (External contours are assigned level 0, internal contours level 1).</td>
</tr>
<tr>
<td>TCVN_CRM_TREE</td>
<td>Returns all contours and their full hierarchy.</td>
</tr>
<tr>
<td>TCVN_CRM_FLOODFILL</td>
<td>Returns the found contours using a floodfill algorithm (only available for DINT images).</td>
</tr>
</tbody>
</table>
Further information

This type specifies how the contour hierarchy is to be taken into account in the contour search.

- **TCVN_CRM_EXTERNAL**
  Only the outer contours are returned; inner contours are not taken into account.

- **TCVN_CRM_LIST**
  All contours are returned. The contour hierarchy is not taken into account.
- **TCVN_CRM_CONNECTED_COMPONENTS**
  All contours are returned, and a 2-level hierarchy is created. In the graphic, all contours at level 1 are shown in red, level 2 is shown in green. It is evident that the next inner contour after a level 2 contour is again on level 1.

- **TCVN_CRM_TREE**
  All contours are returned, and a hierarchy is created based on a tree structure. In the image level 1 is red, level 2 green, level 3 blue and level 4 yellow.

**Related functions**
- `F_VN_FindContoursExp` [1040]
- `F_VN_FindContourHierarchyExp` [1033]

**6.2.4.17  ETcVnContoursMatchComparisonMethod**
Offers comparison methods for contour matching.

**Syntax**

Definition:
TYPE ETcVnContoursMatchComparisonMethod : 
{ 
    TCVN_CMCM_CONTOURS_MATCH_I1 := 1, 
    TCVN_CMCM_CONTOURS_MATCH_I2 := 2, 
    TCVN_CMCM_CONTOURS_MATCH_I3 := 3 
} DINT; 
END_TYPE

### Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_CMCM_CONTOURS_MATCH_I1</td>
<td>Contours are compared using the sum over the differences between the reciprocal individual characteristics.</td>
</tr>
<tr>
<td>TCVN_CMCM_CONTOURS_MATCH_I2</td>
<td>Contours are compared using the sum over the differences between the individual characteristics.</td>
</tr>
<tr>
<td>TCVN_CMCM_CONTOURS_MATCH_I3</td>
<td>Contours are compared using only the maximum difference between the individual characteristics.</td>
</tr>
</tbody>
</table>

### Further information

The available methods are based on Hu invariants. The following algorithms are used, with A representing the first contour and B the second contour.

\[
I_1(A, B) = \sum_{i=1..7} \frac{1}{m_i^A} - \frac{1}{m_i^B} \\
I_2(A, B) = \sum_{i=1..7} m_i^A - m_i^B \\
I_3(A, B) = \max_{i=1..7} m_i^A - m_i^B \\
m_i^A = \text{sign}(h_i^A) \cdot \log(h_i^A) \\
m_i^B = \text{sign}(h_i^B) \cdot \log(h_i^B) \\
\]

\(h_i^A\) and \(h_i^B\) with \(i\) in \([1, 7]\) are each Hu moments of A and B.

### Related functions

- F_VN_MatchContours(Exp) [872]
- F_VN_MatchContours1vsN(Exp) [873]
- F_VN_MatchImageHuMoments [1049]

#### 6.2.4.18 ETcVnDiffusivityTypeKAZE

Offers diffusivity types for feature detection methods KAZE and AKAZE.

### Syntax

Definition:

```
TYPE ETcVnDiffusivityTypeKAZE : 
{ 
    TCVN_DT1_KAZE_PM_G1 := 0, 
    TCVN_DT1_KAZE_PM_G2 := 1, 
    TCVN_DT1_KAZE_WEICKERT := 2, 
} DINT; 
END_TYPE
```
Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_DT1_KAZE_PM_G1</td>
<td>Uses the Perona-Malik diffusivity filter g1.</td>
</tr>
<tr>
<td>TCVN_DT1_KAZE_PM_G2</td>
<td>Uses the Perona-Malik diffusivity filter g2.</td>
</tr>
<tr>
<td>TCVN_DT1_KAZE_WEICKERT</td>
<td>Uses the Weickert diffusivity filter.</td>
</tr>
<tr>
<td>TCVN_DT1_KAZE_CHARBONNIER</td>
<td>Uses the Charbonnier diffusivity filter.</td>
</tr>
</tbody>
</table>

Further information

The enum ETcVnDiffusivityTypeKAZE is used in the structure TcVnParamsKAZE [217].

Related functions

- F_VN_KeyPointsAndDescriptorsKAZEExp [1194]

6.2.4.19 ETcVnDistanceTransformationLabel

Offers types of the label array to build.

Syntax

Definition:

```plaintext
TYPE ETcVnDistanceTransformationLabel :
{  
  TCVN_DTL_CCOMP := 0,
  TCVN_DTL_PIXEL := 1
} DINT;
END_TYPE
```

Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_DTL_CCOMP</td>
<td>Labels connected components (Each connected component of zeros in the source image and the pixels closest to the connected component will be assigned the same label).</td>
</tr>
<tr>
<td>TCVN_DTL_PIXEL</td>
<td>Labels pixels (Each zero pixel and the non-zero pixels closest to it get their own label).</td>
</tr>
</tbody>
</table>

Related functions

- F_VN_DistanceTransformationExp [1065]

6.2.4.20 ETcVnDistanceTransformationMask

Offers different sizes of the distance transformation mask. Please note, that some sizes are not supported by certain distance types.

Syntax

Definition:

```plaintext
TYPE ETcVnDistanceTransformationMask :
{  
  TCVN_DTM_PRECISE := 0,
} DINT;
END_TYPE
```
TCVN_DTM_3 := 3,
TCVN_DTM_5 := 5
END_TYPE

Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_DTM_PRECISE</td>
<td>Uses a precise mask.</td>
</tr>
<tr>
<td>TCVN_DTM_3</td>
<td>Uses a 3x3 mask.</td>
</tr>
<tr>
<td>TCVN_DTM_5</td>
<td>Uses a 5x5 mask.</td>
</tr>
</tbody>
</table>

Related functions

- F_VN_DistanceTransformation(Exp) [1063]

6.2.4.21 ETcVnDistanceType

Offers distance types.

Syntax

Definition:

```plaintext
TYPE ETcVnDistanceType :
{
    TCVN_DT_USER := -1,
    TCVN_DT_L1 := 1,
    TCVN_DT_L2 := 2,
    TCVN_DT_C := 3,
    TCVN_DT_L12 := 4,
    TCVN_DT_FAIR := 5,
    TCVN_DT_WELSCH := 6,
    TCVN_DT_HUBER := 7
} DINT;
END_TYPE
```

Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_DT_USER</td>
<td>User defined distance</td>
</tr>
<tr>
<td>TCVN_DT_L1</td>
<td></td>
</tr>
<tr>
<td>TCVN_DT_L2</td>
<td>Euclidean distance</td>
</tr>
<tr>
<td>TCVN_DT_C</td>
<td>max(</td>
</tr>
<tr>
<td>TCVN_DT_L12</td>
<td>L1 - L2 : 2*(sqrt(1 + x*x/2) - 1))</td>
</tr>
<tr>
<td>TCVN_DT_FAIR</td>
<td>c^2(</td>
</tr>
<tr>
<td>TCVN_DT_WELSCH</td>
<td>c^2/2(1-exp(-x/c)^2)), c = 2.9846</td>
</tr>
<tr>
<td>TCVN_DT_HUBER</td>
<td></td>
</tr>
</tbody>
</table>

Related functions

- F_VN_DistanceTransformation(Exp) [1063]
- F_VN_FitLineExp [870]

6.2.4.22 ETcVnDrawMatchesFlags

Offers a combination of flags to support overdrawing an existing destination image and/or skipping single (i.e. non-matched) keypoints and/or drawing additional (rich-)keypoint information (size and orientation). Used by F_VN_DrawKeypointsExp.
Syntax

Definition:

```plaintext
TYPE ETCvnDrawMatchesFlags :
{
   TCVN_DMFF_DEFAULT := 0,
   TCVN_DMFF_OVERDRAW := 1,
   TCVN_DMFF_SKIPSINGLE := 2,
   TCVN_DMFF_OVERDRAW_SKIPSINGLE := 3,
   TCVN_DMFF_RICHKEYPOINT := 4,
   TCVN_DMFF_OVERDRAW_RICHKEYPOINT := 5,
   TCVN_DMFF_SKIPSINGLE_RICHKEYPOINT := 6,
   TCVN_DMFF_OVERDRAW_SKIPSINGLE_RICHKEYPOINT := 7
}DINT;
END_TYPE
```

Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_DMFF_DEFAULT</td>
<td>Draw all keypoints into a new image.</td>
</tr>
<tr>
<td>TCVN_DMFF_OVERDRAW</td>
<td>Draw all keypoints into the existing destination image.</td>
</tr>
<tr>
<td>TCVN_DMFF_SKIPSINGLE</td>
<td>Draw the keypoints into a new image but skip single keypoints.</td>
</tr>
<tr>
<td>TCVN_DMFF_OVERDRAW_SKIPSINGLE</td>
<td>Draw the keypoints into the existing destination image but skip single keypoints.</td>
</tr>
<tr>
<td>TCVN_DMFF_RICHKEYPOINT</td>
<td>Draw all keypoints with rich information into a new image.</td>
</tr>
<tr>
<td>TCVN_DMFF_OVERDRAW_RICHKEYPOINT</td>
<td>Draw all keypoints with rich information into the existing destination image.</td>
</tr>
<tr>
<td>TCVN_DMFF_SKIPSINGLE_RICHKEYPOINT</td>
<td>Draw the keypoints with rich information into a new image but skip single keypoints.</td>
</tr>
<tr>
<td>TCVN_DMFF_OVERDRAW_SKIPSINGLE_RICHKEYPOINT</td>
<td>Draw the keypoints with rich information into the existing destination image but skip single keypoints.</td>
</tr>
</tbody>
</table>

Related functions

- F V N _ DrawKeypointsExp [{896}]
- F V N _ DrawMatchesExp[908]

6.2.4.23 ETCvNDrawShape

Offers shapes to be drawn.

Syntax

Definition:

```plaintext
TYPE ETCvNDrawShape :
{
   TCVN_DS_RANDOM := -1,
   TCVN_DS_CIRCLE := 0,
   TCVN_DS_SQUARE := 1,
   TCVN_DS_PLUS := 2,
   TCVN_DS_X := 3,
   TCVN_DS_DIAMOND := 4,
   TCVN_DS_MAX := 5
}DINT;
END_TYPE
```
### Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_DS_RANDOM</td>
<td>Randomly select a circle, square, plus, X or diamond.</td>
</tr>
<tr>
<td>TCVN_DS_CIRCLE</td>
<td>Circle</td>
</tr>
<tr>
<td>TCVN_DS_SQUARE</td>
<td>Square</td>
</tr>
<tr>
<td>TCVN_DS_PLUS</td>
<td>+</td>
</tr>
<tr>
<td>TCVN_DS_X</td>
<td>x</td>
</tr>
<tr>
<td>TCVN_DS_DIAMOND</td>
<td>Diamond</td>
</tr>
<tr>
<td>TCVN_DS_MAX</td>
<td>For internal use only, adapted when adding new values.</td>
</tr>
</tbody>
</table>

### Related functions

- F_VN_DrawPoint(Exp) [909]
- F_VN_DrawPoints(Exp) [911]

### 6.2.4.24 ETcVnEdgeDetectionAlgorithm

Offers edge detection algorithms.

#### Syntax

Definition:

```plaintext
TYPE ETcVnEdgeDetectionAlgorithm :
{
    TCVN_EDA_INTERPOLATION := 0,
    TCVN_EDA_APPROX_ERF   := 1,
    TCVN_EDA_APPROX_GAUSSIAN := 2
} DINT;
END_TYPE
```

#### Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_EDA_INTERPOLATION</td>
<td>Interpolates pixels (bilinear) and then finds the maximum gradient. This approach is fast and stable, but usually less precise than the function approximation methods.</td>
</tr>
<tr>
<td>TCVN_EDA_APPROX_ERF</td>
<td>Approximates the edge with an erf function. This approach is slower than the interpolation method, but usually more precise. However, it can be inaccurate if the edge does not suit the erf model.</td>
</tr>
<tr>
<td>TCVN_EDA_APPROX_GAUSSIAN</td>
<td>Approximates the edge with a gaussian function. This method is intended to find the center of relatively thin lines, so it is likely to be inaccurate for other edges.</td>
</tr>
</tbody>
</table>

#### Related functions

- Measurement [1215]
  - F_VN_LocateCircularArcExp [1227]
  - F_VN_LocateEdgeExp [1233]
  - F_VN_LocateEdgesExp [1239]
  - F_VN_LocateEllipseExp [1244]
  - F_VN_MeasureAngleBetweenEdgesExp [1250]
  - F_VN_MeasureEdgeDistanceExp [1255]
6.2.4.25 ETcVnEdgeDirection

Offers edge directions relative to the search direction.

Syntax

Definition:

```plaintext
TYPE ETcVnEdgeDirection :
{
  TCVN_ED_DARK_TO_LIGHT := 0,
  TCVN_ED_LIGHT_TO_DARK := 1
}DINT;
END_TYPE
```

Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_ED_DARK_TO_LIGHT</td>
<td>Dark to light</td>
</tr>
<tr>
<td>TCVN_ED_LIGHT_TO_DARK</td>
<td>Light to dark</td>
</tr>
</tbody>
</table>

Related functions

- Measurement [1215]

6.2.4.26 ETcVnElementType

Offers element types.

Syntax

Definition:

```plaintext
TYPE ETcVnElementType :
{
  TCVN_ET_SAME_AS_SOURCE := -1,
  TCVN_ET_USINT := 0,
  TCVN_ET_SINT := 1,
  TCVN_ET_UINT := 2,
  TCVN_ET_INT := 3,
  TCVN_ET_DINT := 4,
  TCVN_ET_REAL := 5,
  TCVN_ET_LREAL := 6
}DINT;
END_TYPE
```

Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_ET_SAME_AS_SOURCE</td>
<td>Sets the element type of the destination image to the source image element type.</td>
</tr>
<tr>
<td>TCVN_ET_USINT</td>
<td>USINT (depth: 8 bit, 0..255)</td>
</tr>
<tr>
<td>TCVN_ET_SINT</td>
<td>SINT (depth: 8 bit, -128..127)</td>
</tr>
<tr>
<td>TCVN_ET_UINT</td>
<td>UINT (depth: 16 bit, 0..65535)</td>
</tr>
<tr>
<td>TCVN_ET_INT</td>
<td>INT (depth: 16 bit, -32768..32767)</td>
</tr>
<tr>
<td>TCVN_ET_DINT</td>
<td>DINT (depth: 32 bit, -2147483648..2147483647)</td>
</tr>
<tr>
<td>TCVN_ET_REAL</td>
<td>REAL (depth: 32 bit, ~ -3.402823E38 .. ~ 3.402823E38)</td>
</tr>
<tr>
<td>TCVN_ET_LREAL</td>
<td>LREAL (depth: 64 bit, ~ -1.79769313486231E308 .. ~ 1.79769313486232E308)</td>
</tr>
</tbody>
</table>
Further information

The enum `ETcVnElementType` is mainly used to specify the pixel type or bit depth of images. Furthermore, it is also contained in the structure `TCvMatrix`.

Related functions

- `F_VN_ConvertElementType(Exp)`
- `F_VN_CreateImage`
- `F_VN_CreateImageAndSetPixels`
- `F_VN_CreateImageFromArray`
- `F_VN_CustomFilter(Exp)`
- `F_VN_InitMatrixStruct`
- `F_VN_LaplacianFilter(Exp)`
- `F_VN_NormalizeImageExp`
- `F_VN_ScharrFilter(Exp)`
- `F_VN_SepableCustomFilter(Exp)`
- `F_VN_SobelFilter(Exp)`

6.2.4.27  **ETcVnEstimationAlgorithm**

Offers estimation algorithms for matching point sets.

Syntax

Definition:

```plaintext
TYPE ETcVnEstimationAlgorithm :
{
    TCVN_EA_DEFAULT := 0,
    TCVN_EA_LMEDS   := 4,
    TCVN_EA_RANSAC := 8,
    TCVN_EA_RHO    := 16
} DINT;
END_TYPE
```

Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_EA_DEFAULT</td>
<td>Use all points.</td>
</tr>
<tr>
<td>TCVN_EA_LMEDS</td>
<td>Least Median of Squares</td>
</tr>
<tr>
<td>TCVN_EA_RANSAC</td>
<td>Random Sample Consensus</td>
</tr>
<tr>
<td>TCVN_EA_RHO</td>
<td>Progressive RHO Sample Consensus</td>
</tr>
</tbody>
</table>

Related functions

- `F_VN_FindReferenceKeyPointsInImageORBExp`
- `F_VN_HomographyExp`

6.2.4.28  **ETcVnExtremePointDirection**

Offers search directions for the extreme point.
Syntax

Definition:

```plaintext
TYPE EtcVnExtremePointDirection :
{
    TCVN_EPD_TOP_LEFT := 0,
    TCVN_EPD_TOP_MEDIAN := 1,
    TCVN_EPD_TOP_RIGHT := 2,
    TCVN_EPD_BOTTOM_LEFT := 3,
    TCVN_EPD_BOTTOM_MEDIAN := 4,
    TCVN_EPD_BOTTOM_RIGHT := 5,
    TCVN_EPD_LEFT_TOP := 6,
    TCVN_EPD_LEFT_MEDIAN := 7,
    TCVN_EPD_LEFT_BOTTOM := 8,
    TCVN_EPD_RIGHT_TOP := 9,
    TCVN_EPD_RIGHT_MEDIAN := 10,
    TCVN_EPD_RIGHT_BOTTOM := 11
} DINT;
END_TYPE
```

Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_EPD_TOP_LEFT</td>
<td>Find the topmost point (min y, take left one if more than 1).</td>
</tr>
<tr>
<td>TCVN_EPD_TOP_MEDIAN</td>
<td>Find the topmost point (min y, take median one if more than 1).</td>
</tr>
<tr>
<td>TCVN_EPD_TOP_RIGHT</td>
<td>Find the topmost point (min y, take right one if more than 1).</td>
</tr>
<tr>
<td>TCVN_EPD_BOTTOM_LEFT</td>
<td>Find the bottommost point (max y, take left one if more than 1).</td>
</tr>
<tr>
<td>TCVN_EPD_BOTTOM_MEDIAN</td>
<td>Find the bottommost point (max y, take median one if more than 1).</td>
</tr>
<tr>
<td>TCVN_EPD_BOTTOM_RIGHT</td>
<td>Find the bottommost point (max y, take right one if more than 1).</td>
</tr>
<tr>
<td>TCVN_EPD_LEFT_TOP</td>
<td>Find the leftmost point (min x, take top one if more than 1).</td>
</tr>
<tr>
<td>TCVN_EPD_LEFT_MEDIAN</td>
<td>Find the leftmost point (min x, take median one if more than 1).</td>
</tr>
<tr>
<td>TCVN_EPD_LEFT_BOTTOM</td>
<td>Find the leftmost point (min x, take bottom one if more than 1).</td>
</tr>
<tr>
<td>TCVN_EPD_RIGHT_TOP</td>
<td>Find the rightmost point (max x, take top one if more than 1).</td>
</tr>
<tr>
<td>TCVN_EPD_RIGHT_MEDIAN</td>
<td>Find the rightmost point (max x, take median one if more than 1).</td>
</tr>
<tr>
<td>TCVN_EPD_RIGHT_BOTTOM</td>
<td>Find the rightmost point (max x, take bottom one if more than 1).</td>
</tr>
</tbody>
</table>

Related functions

- F_VN_ContourExtremePoint

6.2.4.29 EtcVnFeatureDescriptorTypeAKAZE

Offers descriptor types for AKAZE method.

Syntax

Definition:
**6.2.4.30  ** ETcVnFeatureScoreTypeORB

Offers algorithms used to rank features.

**Syntax**

Definition:

```plaintext
TYPE ETcVnFeatureScoreTypeORB :
{
    TCVN_FST_ORB_HARRIS := 0,
    TCVN_FST_ORB_FAST := 1
}DINT;
END_TYPE
```

**Values**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_FST_ORB_HARRIS</td>
<td>Harris algorithm (produces more stable keypoints, but computation takes longer).</td>
</tr>
<tr>
<td>TCVN_FST_ORB_FAST</td>
<td>FAST algorithm (produces slightly less stable keypoints, but computation is faster).</td>
</tr>
</tbody>
</table>

**Further information**

The enum **ETcVnFeatureScoreTypeORB** is used in the structure **TcVnParamsORB** [p. 218].

**6.2.4.31  ** ETcVnFilterDirection

Offers directions, in which to apply filter.

**Syntax**

Definition:

```plaintext
TYPE ETcVnFilterDirection :
{
    TCVN_FD_X := 0,
    TCVN_FD_Y := 1
}INT;
END_TYPE
```

**Further information**

The enum **ETcVnFilterDirection** is used in the structure **TcVnParams**.
6.2.4.32 ETcVnFlipAxis

Defines the axis around which to flip (mirror) the image.

Syntax

Definition:

```cpp
TYPE ETcVnFlipAxis :
{
   TCVN_FA_XY := -1,
   TCVN_FA_X := 0,
   TCVN_FA_Y := 1
} INT;
END_TYPE
```

Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_FA_XY</td>
<td>Flip the image around both axes</td>
</tr>
<tr>
<td>TCVN_FA_X</td>
<td>Flip the image around the X axis</td>
</tr>
<tr>
<td>TCVN_FA_Y</td>
<td>Flip the image around the Y axis</td>
</tr>
</tbody>
</table>

Related functions

• [F_VN_ScharrFilter(Exp)](#1149)

6.2.4.33 ETcVnFontType

Offers font types.

Syntax

Definition:

```cpp
TYPE ETcVnFontType :
{
   TCVN_FT_HERSHEY_SIMPLEX := 0,
   TCVN_FT_HERSHEY_PLAIN := 1,
   TCVN_FT_HERSHEY_DUPLEX := 2,
   TCVN_FT_HERSHEY_COMPLEX := 3,
   TCVN_FT_HERSHEY_TRIPLEX := 4,
   TCVN_FT_HERSHEY_COMPLEX_SMALL := 5,
   TCVN_FT_HERSHEY_SCRIPT_SIMPLEX := 6,
   TCVN_FT_HERSHEY_SCRIPT_COMPLEX := 7,
   TCVN_FT_HERSHEY_PLAIN_ITALIC := 17,
   TCVN_FT_HERSHEY_COMPLEX_ITALIC := 19,
   TCVN_FT_HERSHEY_TRIPLEX_ITALIC := 20,
   TCVN_FT_HERSHEY_COMPLEX_SMALL_ITALIC := 21
} DINT;
END_TYPE
```

Related functions

• [F_VN_FlipImage](#1001)
## Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_FT_HERSHEY_SIMPLEX</td>
<td>Normal size sans-serif font.</td>
</tr>
<tr>
<td>TCVN_FT_HERSHEY_PLAIN</td>
<td>Small size sans-serif font.</td>
</tr>
<tr>
<td>TCVN_FT_HERSHEY_DUPLEX</td>
<td>More complex normal size sans-serif font.</td>
</tr>
<tr>
<td>TCVN_FT_HERSHEY_COMPLEX</td>
<td>Normal size serif font.</td>
</tr>
<tr>
<td>TCVN_FT_HERSHEY_TRIPLEX</td>
<td>More complex normal size serif font.</td>
</tr>
<tr>
<td>TCVN_FT_HERSHEY_COMPLEX_SMALL</td>
<td>Small size serif font.</td>
</tr>
<tr>
<td>TCVN_FT_HERSHEY_SCRIPT_SIMPLEX</td>
<td>Hand-writing style font.</td>
</tr>
<tr>
<td>TCVN_FT_HERSHEY_SCRIPT_COMPLEX</td>
<td>More complex hand-writing style font.</td>
</tr>
<tr>
<td>TCVN_FT_HERSHEY_PLAIN_ITALIC</td>
<td>Small size sans-serif font (italic).</td>
</tr>
<tr>
<td>TCVN_FT_HERSHEY_COMPLEX_ITALIC</td>
<td>Normal size serif font (italic).</td>
</tr>
<tr>
<td>TCVN_FT_HERSHEY_TRIPLEX_ITALIC</td>
<td>More complex normal size serif font (italic).</td>
</tr>
<tr>
<td>TCVN_FT_HERSHEY_COMPLEX_SMALL_ITALIC</td>
<td>Small size serif font (italic).</td>
</tr>
</tbody>
</table>

## Further information

The following fonts are available:

- **Hershey Simplex**
- **Hershey Plain**
- **Hershey Duplex**
- **Hershey Complex**
- **Hershey Triplex**
- **Hershey Complex Small**
- **Hershey Script Simplex**
- **Hershey Script Complex**

### Plain Italic
- **Complex Italic**
- **Triplex Italic**
- **Complex Small Italic**

## Related functions

- `F_VN_PutLabelExp` [923]
- `F_VN_PutText(Exp)` [924]

## ETcVnHoughMethod

Offers Hough methods.

### Syntax

**Definition:**
TYPE EtcVnHoughMethod :
{
    TCVN_HM_STANDARD := 0,
    TCVN_HM_PROBABILISTIC := 1,
    TCVN_HM_MULTI_SCALE := 2,
    TCVN_HM_GRADIENT := 3,
    TCVN_HM_GRADIENT_ALT := 4
} DINT;
END_TYPE

Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_HM_STANDARD</td>
<td>Classical or standard Hough transform. Every line is represented by two floating-point numbers (f1, f2), where f1 is a distance between (0,0) origin and the line, and f2 is the angle between x-axis and the normal to the line.</td>
</tr>
<tr>
<td>TCVN_HM_PROBABILISTIC</td>
<td>Probabilistic Hough transform. More efficient for pictures containing long linear segments. Returns line segments rather than the whole line, while each segment is represented by start and end point.</td>
</tr>
<tr>
<td>TCVN_HM_MULTI_SCALE</td>
<td>Multi scale variant of classical Hough transform.</td>
</tr>
<tr>
<td>TCVN_HM_GRADIENT_ALT</td>
<td>Variation of the GRADIENT method.</td>
</tr>
</tbody>
</table>

Related functions

- F_VN_HoughCirclesExp

6.2.4.35 ETCvNInterpolationType

Offers interpolation types.

Syntax

Definition:

TYPE ETCvNInterpolationType :
{
    TCVN_IT_NEAREST_NEIGHBOR := 0,
    TCVN_IT_BILINEAR := 1,
    TCVN_IT_BICUBIC := 2,
    TCVN_IT_AREA_BASED := 3,
    TCVN_IT_LANCZOS4 := 4
} DINT;
END_TYPE

Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_IT_NEAREST_NEIGHBOR</td>
<td>Nearest neighbor interpolation</td>
</tr>
<tr>
<td>TCVN_IT_BILINEAR</td>
<td>Bilinear interpolation</td>
</tr>
<tr>
<td>TCVN_IT_BICUBIC</td>
<td>Bicubic interpolation</td>
</tr>
<tr>
<td>TCVN_IT_AREA_BASED</td>
<td>Area based interpolation</td>
</tr>
<tr>
<td>TCVN_IT_LANCZOS4</td>
<td>Lanczos4 interpolation</td>
</tr>
</tbody>
</table>
Related functions

- F_VN_MatchTemplateAndEvaluateExp \[\text{1052}\]
- F_VN_RemapImageToLogPolarSpaceExp \[\text{1013}\]
- F_VN_RemapImageToPolarSpaceExp \[\text{1014}\]
- F_VN_Resizelmage \[\text{1015}\]
- F_VN_WarpAffineExp \[\text{1020}\]
- F_VN_WarpPerspectiveExp \[\text{1025}\]

6.2.4.36 \textbf{ETcVnKeypointDetectionTypeAGAST}

Offers different neighborhood types for AGAST method (For details see paper: E. Mair et al.: Adaptive and Generic Corner Detection Based on the Accelerated Segment Test, 2010).

Syntax

Definition:

```plaintext
TYPE ETcVnKeypointDetectionTypeAGAST :
{
    TCVN_KDT_AGAST_5_8 := 0,
    TCVN_KDT_AGAST_7_12d := 1,
    TCVN_KDT_AGAST_7_12s := 2,
    TCVN_KDT_AGAST_9_16 := 3
}DINT;
END_TYPE
```

Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_KDT_AGAST_5_8</td>
<td>5 of 8</td>
</tr>
<tr>
<td>TCVN_KDT_AGAST_7_12d</td>
<td>7 of 12 diamond</td>
</tr>
<tr>
<td>TCVN_KDT_AGAST_7_12s</td>
<td>7 of 12 square</td>
</tr>
<tr>
<td>TCVN_KDT_AGAST_9_16</td>
<td>9 of 16</td>
</tr>
</tbody>
</table>

Further information

The enum ETcVnKeypointDetectionTypeAGAST is used in the structure TcVnParamsAGAST \[\text{210}\].

6.2.4.37 \textbf{ETcVnKeypointDetectionTypeFAST}

Offers different neighborhood types for FAST method (For details see paper: E. Rosten: Machine Learning for High-speed Corner Detection, 2006).

Syntax

Definition:

```plaintext
TYPE ETcVnKeypointDetectionTypeFAST :
{
    TCVN_KDT_FAST_5_8 := 0,
    TCVN_KDT_FAST_7_12 := 1,
    TCVN_KDT_FAST_9_16 := 2
}DINT;
END_TYPE
```
Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_KDT_FAST_5_8</td>
<td>5 of 8</td>
</tr>
<tr>
<td>TCVN_KDT_FAST_7_12</td>
<td>7 of 12</td>
</tr>
<tr>
<td>TCVN_KDT_FAST_9_16</td>
<td>9 of 16</td>
</tr>
</tbody>
</table>

Further information

The enum `ETcVnKeypointDetectionTypeFAST` is used in the structure `TcVnParamsFAST [216].`

6.2.4.38  ETcVnLineType

Offers line types.

Syntax

Definition:
```plaintext
TYPE ETcVnLineType :
{
   TCVN_LT_4_CONNECTED := 4,
   TCVN_LT_8_CONNECTED := 8,
   TCVN_LT_ANTIALIASED := 16
} DINT;
END_TYPE
```

Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_LT_4_CONNECTED</td>
<td>4-connected (pixels are connected horizontally and vertically).</td>
</tr>
<tr>
<td>TCVN_LT_8_CONNECTED</td>
<td>8-connected (pixels are connected horizontally, vertically and diagonally).</td>
</tr>
<tr>
<td>TCVN_LT_ANTIALIASED</td>
<td>Antialiased (drawn using Gaussian filtering, only implemented for 8-bit images).</td>
</tr>
</tbody>
</table>

Related functions

- F_VN_DrawCircleExp [885]
- F_VN_DrawCirclesExp [887]
- F_VN_DrawCircularArcExp [889]
- F_VN_DrawContoursExp [892]
- F_VN_DrawEllipseExp [895]
- F_VN_DrawLineExp [901]
- F_VN_DrawLineExp_TcVnVector4_DINT [902]
- F_VN_DrawLineExp_TcVnVector4_LREAL [903]
- F_VN_DrawLinesExp [905]
- F_VN_DrawPointExp [910]
- F_VN_DrawPointsExp [912]
- F_VN_DrawRotatedRectangleExp [916]
- F_VN_PutLabelExp [923]
- F_VN_PutTextExp [926]
6.2.4.39  ETcVnMorphologicalOperator

Offers morphological operators.

Syntax

Definition:

```plaintext
TYPE ETcVnMorphologicalOperator :
{
  TCVN_MO_EROSION := 0x0,
  TCVN_MO_DILATION := 0x1,
  TCVN_MO_OPENING := 0x2,
  TCVN_MO_CLOSING := 0x3,
  TCVN_MO_GRADIENT := 0x4,
  TCVN_MO_WHITE_TOPHAT := 0x5,
  TCVN_MO_BLACK_TOPHAT := 0x6,
  TCVN_MO_OPENING_BY_RECONSTRUCTION := 0x40000002,
  TCVN_MO_CLOSING_BY_RECONSTRUCTION := 0x40000003,
  TCVN_MO_WHITE_TOPHAT_BY_RECONSTRUCTION := 0x40000005,
  TCVN_MO_BLACK_TOPHAT_BY_RECONSTRUCTION := 0x40000006
)DINT;
END_TYPE
```

Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_MO_EROSION</td>
<td>Shrinks objects (regions of foreground (i.e. white) pixels), removing regions smaller than the structuring element completely.</td>
</tr>
<tr>
<td>TCVN_MO_DILATION</td>
<td>Expands objects and thereby closes small holes inside objects.</td>
</tr>
<tr>
<td>TCVN_MO_OPENING</td>
<td>Applies an erosion first, a dilation second. Objects smaller than the structuring element are removed while outer shapes remain largely the same.</td>
</tr>
<tr>
<td>TCVN_MO_CLOSING</td>
<td>Applies a dilation first, an erosion second. Holes inside objects that fit into the structuring element are closed completely while outer shapes remain largely the same.</td>
</tr>
<tr>
<td>TCVN_MO_GRADIENT</td>
<td>Difference between the dilation and the erosion of an image.</td>
</tr>
<tr>
<td>TCVN_MO_WHITE_TOPHAT</td>
<td>Difference between an input image and its opening.</td>
</tr>
<tr>
<td>TCVN_MO_BLACK_TOPHAT</td>
<td>Difference between an input image and its closing.</td>
</tr>
<tr>
<td>TCVN_MO_OPENING_BY_RECONSTRUCTION</td>
<td>Opening with subsequent reconstruction of objects that were not removed by the opening.</td>
</tr>
<tr>
<td>TCVN_MO_CLOSING_BY_RECONSTRUCTION</td>
<td>Closing with subsequent reconstruction of objects that were not removed by the closing.</td>
</tr>
<tr>
<td>TCVN_MO_WHITE_TOPHAT_BY_RECONSTRUCTION</td>
<td>White tophat with subsequent reconstruction of objects that were not removed by the white tophat.</td>
</tr>
<tr>
<td>TCVN_MO_BLACK_TOPHAT_BY_RECONSTRUCTION</td>
<td>Black tophat with subsequent reconstruction of objects that were not removed by the black tophat.</td>
</tr>
</tbody>
</table>

Related functions

- F_VN_MorphologicalOperator [1145]

6.2.4.40  ETcVnNormalizationType

Offers normalization types.
Syntax

Definition:

```plaintext
TYPE ETcVnNormalizationType :
{
    TCVN_NT_INF := 1,
    TCVN_NT_L1  := 2,
    TCVN_NT_L2  := 4,
    TCVN_NT_L2SQR := 5,
    TCVN_NT_HAMMING := 6,
    TCVN_NT_HAMMING2 := 7,
    TCVN_NT_RELATIVE_INF := 9,
    TCVN_NT_RELATIVE_L1 := 10,
    TCVN_NT_RELATIVE_L2 := 12,
    TCVN_NT_RELATIVE_L2SQR := 13,
    TCVN_NT_RELATIVE_HAMMING := 14,
    TCVN_NT_RELATIVE_HAMMING2 := 15,
    TCVN_NT_MINMAX := 32
} DINT;
END_TYPE
```

Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_NT_INF</td>
<td>Infinity norm</td>
</tr>
<tr>
<td>TCVN_NT_L1</td>
<td>L1 norm</td>
</tr>
<tr>
<td>TCVN_NT_L2</td>
<td>L2 (euclidean) norm</td>
</tr>
<tr>
<td>TCVN_NT_L2SQR</td>
<td>Squared L2 norm</td>
</tr>
<tr>
<td>TCVN_NT_HAMMING</td>
<td>Hamming distance (bitwise)</td>
</tr>
<tr>
<td>TCVN_NT_HAMMING2</td>
<td>Hamming distance (each 2 bits combined to one)</td>
</tr>
<tr>
<td>TCVN_NT_RELATIVE_INF</td>
<td>Infinity norm (when comparing 2 arrays, the norm of their difference is divided by the norm of the second array)</td>
</tr>
<tr>
<td>TCVN_NT_RELATIVE_L1</td>
<td>L1 norm (when comparing 2 arrays, the norm of their difference is divided by the norm of the second array)</td>
</tr>
<tr>
<td>TCVN_NT_RELATIVE_L2</td>
<td>L2 (euclidean) norm (when comparing 2 arrays, the norm of their difference is divided by the norm of the second array)</td>
</tr>
<tr>
<td>TCVN_NT_RELATIVE_L2SQR</td>
<td>Squared L2 norm (when comparing 2 arrays, the norm of their difference is divided by the norm of the second array)</td>
</tr>
<tr>
<td>TCVN_NT_RELATIVE_HAMMING</td>
<td>Hamming distance (bitwise; when comparing 2 arrays, the norm of their difference is divided by the norm of the second array)</td>
</tr>
<tr>
<td>TCVN_NT_RELATIVE_HAMMING2</td>
<td>Hamming distance (each 2 bits combined to one; when comparing 2 arrays, the norm of their difference is divided by the norm of the second array)</td>
</tr>
<tr>
<td>TCVN_NT_MINMAX</td>
<td>Normalize the values to a range given by a minimum and a maximum value.</td>
</tr>
</tbody>
</table>

Related functions

- F_VN_FindReferenceKeyPointsInImageORBExp [1185]
- F_VN_MatchDescriptorsBFExp [1206]
- F_VN_MatchDescriptorsKnnBFExp [1210]
- F_VN_NormalizeImageExp [1095]

6.2.4.41 ETcVnPixelConnectivity

Offers pixel connectivities.
Syntax

Definition:

```plaintext
TYPE ETcVnPixelConnectivity :
{
    TCVN_PC_4 := 4,
    TCVN_PC_8 := 8
} DINT;
END_TYPE
```

Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_PC_4</td>
<td>4-way</td>
</tr>
<tr>
<td>TCVN_PC_8</td>
<td>8-way</td>
</tr>
</tbody>
</table>

### 6.2.4.42 ETcVnPixelEncoding

Offers pixel encodings.

Syntax

Definition:

```plaintext
TYPE ETcVnPixelEncoding :
{
    TCVN_PE_NONE := 0,
    TCVN_PE_BAYER_GR := 1,
    TCVN_PE_BAYER_RG := 2,
    TCVN_PE_BAYER_GB := 3,
    TCVN_PE_BAYER_BG := 4,
    TCVN_PE_YUV_411_UYYVYY := 5,
    TCVN_PE_YUV_422_UYVV := 6,
    TCVN_PE_YUV_422_YUYV := 7,
    TCVN_PE_YCBCR_411_CBYYCRYY := 8,
    TCVN_PE_YCBCR_422_CBYCRY := 9,
    TCVN_PE_YCBCR_422_YCBYCR := 10
} BYTE;
END_TYPE
```

Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_PE_NONE</td>
<td>No encoding available, i.e. every pixel value is independent of other pixels.</td>
</tr>
<tr>
<td>TCVN_PE_BAYER_GR</td>
<td>Pixels are encoded as a BayerGR pattern.</td>
</tr>
<tr>
<td>TCVN_PE_BAYER_RG</td>
<td>Pixels are encoded as a BayerRG pattern.</td>
</tr>
<tr>
<td>TCVN_PE_BAYER_GB</td>
<td>Pixels are encoded as a BayerGB pattern.</td>
</tr>
<tr>
<td>TCVN_PE_BAYER_BG</td>
<td>Pixels are encoded as a BayerBG pattern.</td>
</tr>
<tr>
<td>TCVN_PE_YUV_411_UYYVYY</td>
<td>Pixels are encoded as YUV411 (UYYVYY).</td>
</tr>
<tr>
<td>TCVN_PE_YUV_422_UYVV</td>
<td>Pixels are encoded as YUV422 (UYVV).</td>
</tr>
<tr>
<td>TCVN_PE_YUV_422_YUYV</td>
<td>Pixels are encoded as YUV422 (YUYV).</td>
</tr>
<tr>
<td>TCVN_PE_YCBCR_411_CBYYCRYY</td>
<td>Pixels are encoded as YCbCr411 (CbYYCrYY).</td>
</tr>
<tr>
<td>TCVN_PE_YCBCR_422_CBYCRY</td>
<td>Pixels are encoded as YCbCr422 (CbYCrY).</td>
</tr>
<tr>
<td>TCVN_PE_YCBCR_422_YCBYCR</td>
<td>Pixels are encoded as YCbCr422 (YCbYCr).</td>
</tr>
</tbody>
</table>

Further information

The enum `ETcVnPixelEncoding` is used in the structure `TcVnPixelFormat` [221]. Information about the pixel encoding can be obtained by querying the pixel format information of an image. A frequent use case is the query for a Bayer pattern in order to find out whether a picture still needs to be converted into an RGB format, for example.
6.2.4.43 ETcVnPixelFormat

Offers pixel packing modes.

Syntax

Definition:

```plaintext
TYPE ETcVnPixelFormat :
{
    TCVN_PPM_NONE := 0,
    TCVN_PPM_MONO1P := 1,
    TCVN_PPM_MONO2P := 2,
    TCVN_PPM_MONO4P := 3,
    TCVN_PPM_MONO10PACKED := 4,
    TCVN_PPM_MONO12PACKED := 5,
    TCVN_PPM_RGB10V1PACKED := 6,
    TCVN_PPM_RGB10P32 := 7,
    TCVN_PPM_RGB12V1PACKED := 8,
    TCVN_PPM_RGB565P := 9,
    TCVN_PPM_BGR565P := 10,
    TCVN_PPM_MONO10P := 11,
    TCVN_PPM_MONO12P := 12
}DINT;
END_TYPE
```

Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_PPM_NONE</td>
<td>No packing</td>
</tr>
<tr>
<td>TCVN_PPM_MONO1P</td>
<td>Mono1p</td>
</tr>
<tr>
<td>TCVN_PPM_MONO2P</td>
<td>Mono2p</td>
</tr>
<tr>
<td>TCVN_PPM_MONO4P</td>
<td>Mono4p</td>
</tr>
<tr>
<td>TCVN_PPM_MONO10PACKED</td>
<td>Mono10Packed or BayerXX10Packed</td>
</tr>
<tr>
<td>TCVN_PPM_MONO12PACKED</td>
<td>Mono12Packed or BayerXX12Packed</td>
</tr>
<tr>
<td>TCVN_PPM_RGB10V1PACKED</td>
<td>RGB10V1Packed</td>
</tr>
<tr>
<td>TCVN_PPM_RGB10P32</td>
<td>RGB10V2Packed (RGB10p32)</td>
</tr>
<tr>
<td>TCVN_PPM_RGB12V1PACKED</td>
<td>RGB12V1Packed</td>
</tr>
<tr>
<td>TCVN_PPM_RGB565P</td>
<td>RGB565p</td>
</tr>
<tr>
<td>TCVN_PPM_BGR565P</td>
<td>BGR565p</td>
</tr>
<tr>
<td>TCVN_PPM_MONO10P</td>
<td>Mono10P or BayerXX10p</td>
</tr>
<tr>
<td>TCVN_PPM_MONO12P</td>
<td>Mono12P or BayerXX12p</td>
</tr>
</tbody>
</table>

Further information

The enum ETcVnPixelFormat is used in the structure TcVnPixelFormat [221].

6.2.4.44 ETcVnRectangleIntersection

Offers rectangle intersection types.

Syntax

Definition:

```plaintext
TYPE ETcVnRectangleIntersection :
{
    TCVN_RI_NONE := 0,
    TCVN_RI_PARTIAL := 1,
    TCVN_RI_FULL := 2
}DINT;
END_TYPE
```
### Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_RI_NONE</td>
<td>No intersection</td>
</tr>
<tr>
<td>TCVN_RI_PARTIAL</td>
<td>Partial intersection</td>
</tr>
<tr>
<td>TCVN_RI_FULL</td>
<td>Full intersection</td>
</tr>
</tbody>
</table>

#### 6.2.4.45 \( \text{ETcVnRotationAngle} \)

Offers rotation angles.

**Syntax**

**Definition:**

```plaintext
TYPE ETcVnRotationAngle :
{
    TCVN_RA_90_DEG := 0,
    TCVN_RA_180_DEG := 1,
    TCVN_RA_270_DEG := 2

) DINT;
END_TYPE
```

**Values**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_RA_90_DEG</td>
<td>90 degrees</td>
</tr>
<tr>
<td>TCVN_RA_180_DEG</td>
<td>180 degrees</td>
</tr>
<tr>
<td>TCVN_RA_270_DEG</td>
<td>270 degrees</td>
</tr>
</tbody>
</table>

**Related functions**

- `F_VN_RotateImage`

#### 6.2.4.46 \( \text{ETcVnSignedNormalization} \)

Offers options for normalizing images with signed data types.

**Syntax**

**Definition:**

```plaintext
TYPE ETcVnSignedNormalization :
{
    TCVN_SN_FIX_ZERO := 0,
    TCVN_SN_FULL_SCALE := 1

) INT;
END_TYPE
```

**Values**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_SN_FIX_ZERO</td>
<td>The value 0 is fixed, i.e. the normalized image might only reach either the minimum or maximum value.</td>
</tr>
<tr>
<td>TCVN_SN_FULL_SCALE</td>
<td>The image is normalized to the full value range, i.e. a prior value of 0 might be ! = 0 after normalization.</td>
</tr>
</tbody>
</table>

**Related functions**

- `F_VN_NormalizeImageForDisplay`
6.2.4.47   **ETcVnStructuringElementShape**

**Offers shapes for a structuring element.**

**Syntax**

**Definition:**

```plaintext
TYPE ETcVnStructuringElementShape :
{
    TCVN_SES_RECTANGLE := 0,
    TCVN_SES_CROSS      := 1,
    TCVN_SES_ELLIPSE    := 2
}DINT;
END_TYPE
```

**Values**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_SES_RECTANGLE</td>
<td>Rectangle</td>
</tr>
<tr>
<td>TCVN_SES_CROSS</td>
<td>Cross</td>
</tr>
<tr>
<td>TCVN_SES_ELLIPSE</td>
<td>Ellipse</td>
</tr>
</tbody>
</table>

**Related functions**

- F_VN_CreateStructuringElement [1121]

6.2.4.48   **ETcVnTemplateMatchMethod**

**Offers methods for template matching.**

**Syntax**

**Definition:**

```plaintext
TYPE ETcVnTemplateMatchMethod :
{
    TCVN_TMM_SQDIFF        := 0,
    TCVN_TMM_SQDIFF_NORMED := 1,
    TCVN_TMM_CCORR         := 2,
    TCVN_TMM_CCORR_NORMED  := 3,
    TCVN_TMM_CCOEFF        := 4,
    TCVN_TMM_CCOEFF_NORMED := 5
}DINT;
END_TYPE
```

**Values**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_TMM_SQDIFF</td>
<td>Squared difference (supports template mask)</td>
</tr>
<tr>
<td>TCVN_TMM_SQDIFF_NORMED</td>
<td>Normalized squared difference</td>
</tr>
<tr>
<td>TCVN_TMM_CCORR</td>
<td>Cross-correlation</td>
</tr>
<tr>
<td>TCVN_TMM_CCORR_NORMED</td>
<td>Normalized cross-correlation (supports template mask)</td>
</tr>
<tr>
<td>TCVN_TMM_CCOEFF</td>
<td>Correlation coefficient</td>
</tr>
<tr>
<td>TCVN_TMM_CCOEFF_NORMED</td>
<td>Normalized correlation coefficient</td>
</tr>
</tbody>
</table>

**Related functions**

- F_VN_MatchTemplateAndEvaluateExp [1052]
- F_VN_MatchTemplateExp [1053]
6.2.4.49   ETcVnThresholdType

Offers threshold types.

Syntax

Definition:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_TT_BINARY</td>
<td>Binary threshold</td>
</tr>
<tr>
<td>TCVN_TT_BINARY_INV</td>
<td>Inverted binary threshold</td>
</tr>
<tr>
<td>TCVN_TT_TRUNC</td>
<td>Truncated threshold (pixels &gt; thresh are set to thresh, others keep their value)</td>
</tr>
<tr>
<td>TCVN_TT_TOZERO</td>
<td>To zero threshold (pixels &lt; thresh are set to zero, others keep their value)</td>
</tr>
<tr>
<td>TCVN_TT_TOZERO_INV</td>
<td>Inverted to zero threshold (pixels &gt; thresh are set to zero, others keep their value)</td>
</tr>
<tr>
<td>TCVN_TT_OOTSU_BINARY</td>
<td>Binary threshold with the threshold value selected according to the Otsu algorithm</td>
</tr>
<tr>
<td>TCVN_TT_OOTSU_BINARY_INV</td>
<td>Inverted binary threshold with the threshold value selected according to the Otsu algorithm</td>
</tr>
<tr>
<td>TCVN_TT_OOTSU_TRUNC</td>
<td>Truncated threshold with the threshold value selected according to the Otsu algorithm</td>
</tr>
<tr>
<td>TCVN_TT_OOTSU_TOZERO</td>
<td>To zero threshold with the threshold value selected according to the Otsu algorithm</td>
</tr>
<tr>
<td>TCVN_TT_OOTSU_TOZERO_INV</td>
<td>Inverted to zero threshold with the threshold value selected according to the Otsu algorithm</td>
</tr>
<tr>
<td>TCVN_TT_TRIANGLE_BINARY</td>
<td>Binary threshold with the threshold value selected according to the Triangle algorithm</td>
</tr>
<tr>
<td>TCVN_TT_TRIANGLE_BINARY_INV</td>
<td>Inverted binary threshold with the threshold value selected according to the Triangle algorithm</td>
</tr>
<tr>
<td>TCVN_TT_TRIANGLE_TRUNC</td>
<td>Truncated threshold with the threshold value selected according to the Triangle algorithm</td>
</tr>
<tr>
<td>TCVN_TT_TRIANGLE_TOZERO</td>
<td>To zero threshold with the threshold value selected according to the Triangle algorithm</td>
</tr>
<tr>
<td>TCVN_TT_TRIANGLE_TOZERO_INV</td>
<td>Inverted to zero threshold with the threshold value selected according to the Triangle algorithm</td>
</tr>
</tbody>
</table>
Further information

The Enum ETcVnThresholdType offers a choice of the methods implemented to calculate the threshold value. It is also used in the structure TcVnParamsBlobDetection [211].

Related functions

- F_VN_AdaptiveThresholdExp [1163]
- F_VN_Threshold [1166]

6.2.4.50 ETcVnVectorCompareMethod

Offers methods for vector comparison.

Syntax

Definition:

```plaintext
TYPE ETcVnVectorCompareMethod :

  (TCVN_VCM_EUCLIDEAN := 0,
   TCVN_VCM_ELEMENTWISE := 1)
DINT;
END_TYPE
```

Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_VCM_EUCLIDEAN</td>
<td>Euclidean</td>
</tr>
<tr>
<td>TCVN_VCM_ELEMENTWISE</td>
<td>Elementwise</td>
</tr>
</tbody>
</table>

Related functions

- F_VN_MaxPixelValueExp [1075]
- F_VN_MinPixelValueExp [1077]

6.2.4.51 ETcWatchdogAccumulationType

Offers watchdog accumulation types to compute the fraction processed.

Syntax

Definition:

```plaintext
TYPE ETcWatchdogAccumulationType :

  (WD_ACC_TYPE_MEAN := 1,
   WD_ACC_TYPE_PRODUCT := 2)
UINT;
END_TYPE
```

Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WD_ACC_TYPE_MEAN</td>
<td>Calculates the mean of the individual fractions processed over all monitored functions (recommended for independent functions).</td>
</tr>
<tr>
<td>WD_ACC_TYPE_PRODUCT</td>
<td>Calculates the product of the individual fractions processed over all monitored functions (recommended for dependent functions).</td>
</tr>
</tbody>
</table>
Further information

With this enum you can set how a watchdog[137] calculates the fraction of the executed functions (nFractionProcessed) when several functions are concatenated.

As an example to illustrate the mode of operation, it is assumed that three functions were monitored. The first was executed 100%, the second 50% and the third 0%.

| WD_ACC_TYPE_MEAN | The average value of the execution fractions of all functions is returned as the result. In the example case the result would be: \((100\% + 50\% + 0\%) / 3 = 50\%\). |
| WD_ACC_TYPE_PRODUCT | The product of the execution fractions of all functions is returned as the result. In the example case the result would be: \(100\% \times 50\% \times 0\% = 0\%\). |

The use cases differ depending on the program structure and the safe states of the machine.

Related functions

- F_VN_StartAbsWatchdogExp[879]
- F_VN_StartRelWatchdogExp[881]

6.2.4.52 GEV_CAMERA_STATE

Describes the GigEVision camera connection state.

Syntax

Definition:

```plaintext
TYPE GEV_CAMERA_STATE :
{
    GEV_CAMERA_IDLE := 0,
    GEV_CONTROL_CHANNEL_OPEN_MASK := 1,
    GEV_STREAM_CHANNELS_OPEN_MASK := 2,
    GEV_CONTROL_CHANNEL_OPEN := 1,
    GEV_STREAM_CHANNELS_OPEN := 3
} BYTE;
END_TYPE
```

Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GEV_CAMERA_IDLE</td>
<td>Camera is idle (no control or stream channel open).</td>
</tr>
<tr>
<td>GEV_CONTROL_CHANNEL_OPEN_MASK</td>
<td>The control channel is open.</td>
</tr>
<tr>
<td>GEV_STREAM_CHANNELS_OPEN_MASK</td>
<td>At least 1 stream channel is open.</td>
</tr>
<tr>
<td>GEV_CONTROL_CHANNEL_OPEN</td>
<td>The control channel is open, but no stream channel.</td>
</tr>
<tr>
<td>GEV_STREAM_CHANNELS_OPEN</td>
<td>At least 1 stream channel and the control channel are open.</td>
</tr>
</tbody>
</table>

6.2.5 Structs

6.2.5.1 GVSP_IMAGE_INFO

Shows GVSP (GigE Vision Streaming Protocol) meta information.
Syntax

Definition:

```
TYPE GVSP_IMAGE_INFO :
  STRUCT
    CameraIpAddress : UDINT;
    LocalIpAddress : UDINT;
    CameraUdpPort : UINT;
    LocalUdpPort : UINT;
    GvspChannelId : UINT;
    GevStatus : UINT;
    BlockId : ULINT;
    LeaderInfo : GVSP_LEADER_PAYLOAD_IMAGE;
  END_STRUCT
  END_TYPE
```

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CameraIpAddress</td>
<td>UDINT</td>
<td>Camera IP address</td>
</tr>
<tr>
<td>LocalIpAddress</td>
<td>UDINT</td>
<td>Local IP address</td>
</tr>
<tr>
<td>CameraUdpPort</td>
<td>UINT</td>
<td>Camera UDP port</td>
</tr>
<tr>
<td>LocalUdpPort</td>
<td>UINT</td>
<td>Local UDP port</td>
</tr>
<tr>
<td>GvspChannelId</td>
<td>UINT</td>
<td>ID of GVSP channel</td>
</tr>
<tr>
<td>GevStatus</td>
<td>UINT</td>
<td>GigE Vision status code</td>
</tr>
<tr>
<td>BlockId</td>
<td>ULINT</td>
<td>Block Id (incremented for each acquired image, but reset to 0 on overflow)</td>
</tr>
<tr>
<td>LeaderInfo</td>
<td>GVSP_LEADER_PAYLOAD_IMAGE</td>
<td>Contains information about timestamp, pixel format and size.</td>
</tr>
</tbody>
</table>

6.2.5.2 GVSP_LEADER_PAYLOAD_IMAGE

Shows information over the GVSP (GigE Vision Streaming Protocol) leader payload image.

Syntax

Definition:

```
TYPE GVSP_LEADER_PAYLOAD_IMAGE :
  STRUCT
    Timestamp : ULINT;
    PixelFormat : GVSP_PIXEL_FORMAT;
    SizeX : UDINT;
    SizeY : UDINT;
    OffsetX : UDINT;
    OffsetY : UDINT;
    PaddingX : UINT;
    PaddingY : UINT;
  END_STRUCT
  END_TYPE
```
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamp</td>
<td>ULINT</td>
<td>Image timestamp</td>
</tr>
<tr>
<td>PixelFormat</td>
<td>GVSP_PIXEL_FORMAT[0x205]</td>
<td>Image pixel format</td>
</tr>
<tr>
<td>SizeX</td>
<td>UDINT</td>
<td>Image size in x direction</td>
</tr>
<tr>
<td>SizeY</td>
<td>UDINT</td>
<td>Image size in y direction</td>
</tr>
<tr>
<td>OffsetX</td>
<td>UDINT</td>
<td>Image x-offset from (0,0) origin</td>
</tr>
<tr>
<td>OffsetY</td>
<td>UDINT</td>
<td>Image y-offset from (0,0) origin</td>
</tr>
<tr>
<td>PaddingX</td>
<td>UINT</td>
<td>Image padding in x direction</td>
</tr>
<tr>
<td>PaddingY</td>
<td>UINT</td>
<td>Image padding in y direction</td>
</tr>
</tbody>
</table>

6.2.5.3  GVSP_PIXEL_FORMAT

Shows the GVSP (GigE Vision Streaming Protocol) pixel format.

Syntax

Definition:

```
TYPE GVSP_PIXEL_FORMAT :
  STRUCT
    Color : BYTE;
    EffectivePixelSize : BYTE;
    Id : UINT;
  END_STRUCT
END_TYPE
```

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td>BYTE</td>
<td>Indicates if the pixel format is mono or color.</td>
</tr>
<tr>
<td>EffectivePixelSize</td>
<td>BYTE</td>
<td>Effective pixel size in bit</td>
</tr>
<tr>
<td>Id</td>
<td>UINT</td>
<td>GVSP pixel format ID</td>
</tr>
</tbody>
</table>

6.2.5.4  TcVnCameraCalibrationOptions

Offers camera calibration options.

Syntax

Definition:

```
TYPE TcVnCameraCalibrationOptions :
  STRUCT
    bUseIntrinsicGuess : BIT;
    bFixAspectRatio    : BIT;
    bFixPrincipalPoint : BIT;
    bZeroTangentDist   : BIT;
    bFixFocalLength    : BIT;
    bFixK1             : BIT;
    bFixK2             : BIT;
    bFixK3             : BIT;
    bFixK4             : BIT;
    bFixK5             : BIT;
    bFixK6             : BIT;
    bRationalModel     : BIT;
  END_STRUCT
END_TYPE
```
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bUseIntrinsicGuess</td>
<td>BIT</td>
<td>The camera matrix and distortion coefficients already contain a valid initial guess, which is optimized further.</td>
</tr>
<tr>
<td>bFixAspectRatio</td>
<td>BIT</td>
<td>The ratio fx/fy stays the same as in the input camera matrix.</td>
</tr>
<tr>
<td>bFixPrincipalPoint</td>
<td>BIT</td>
<td>The principal point is fixed to the image center (or provided cx, cy if bUseIntrinsicGuess).</td>
</tr>
<tr>
<td>bZeroTangentDist</td>
<td>BIT</td>
<td>The tangential distortion coefficients (p1, p2) are forced to 0.</td>
</tr>
<tr>
<td>bFixFocalLength</td>
<td>BIT</td>
<td>The parameters fx and fy stay the same as in the input camera matrix.</td>
</tr>
<tr>
<td>bFixK1</td>
<td>BIT</td>
<td>The radial distortion coefficient k1 is fixed to 0 (or provided input if bUseIntrinsicGuess).</td>
</tr>
<tr>
<td>bFixK2</td>
<td>BIT</td>
<td>The radial distortion coefficient k2 is fixed to 0 (or provided input if bUseIntrinsicGuess).</td>
</tr>
<tr>
<td>bFixK3</td>
<td>BIT</td>
<td>The radial distortion coefficient k3 is fixed to 0 (or provided input if bUseIntrinsicGuess).</td>
</tr>
<tr>
<td>bFixK4</td>
<td>BIT</td>
<td>The radial distortion coefficient k4 is fixed to 0 (or provided input if bUseIntrinsicGuess).</td>
</tr>
<tr>
<td>bFixK5</td>
<td>BIT</td>
<td>The radial distortion coefficient k5 is fixed to 0 (or provided input if bUseIntrinsicGuess).</td>
</tr>
<tr>
<td>bFixK6</td>
<td>BIT</td>
<td>The radial distortion coefficient k6 is fixed to 0 (or provided input if bUseIntrinsicGuess).</td>
</tr>
<tr>
<td>bRationalModel</td>
<td>BIT</td>
<td>The radial distortion coefficients k4, k5, k6 are enabled.</td>
</tr>
</tbody>
</table>

### 6.2.5.5 TcVnCircularArc

Describes a circular arc.

**Syntax**

**Definition:**

```plaintext
TYPE TcVnCircularArc :
  STRUCT
    aCenter : TcVnPoint2_REAL;
    fRadius : REAL;
    fStartAngle : REAL;
    fEndAngle : REAL;
  END_STRUCT
END_TYPE
```
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aCenter</td>
<td>TcVnPoint2_REAL[151]</td>
<td>Center of the circular arc</td>
</tr>
<tr>
<td>fRadius</td>
<td>REAL</td>
<td>Radius of the circular arc</td>
</tr>
<tr>
<td>fStartAngle</td>
<td>REAL</td>
<td>Start angle of the circular arc</td>
</tr>
<tr>
<td>fEndAngle</td>
<td>REAL</td>
<td>End angle of the circular arc</td>
</tr>
</tbody>
</table>

#### 6.2.5.6 TcVnDMatch

Describes a descriptor match.

**Syntax**

**Definition:**

```c
TYPE TcVnDMatch :
STRUCT
    nQueryIdx : DINT;
    nTrainIdx : DINT;
    nImageIdx : DINT;
    fDistance : REAL;
END_STRUCT
END_TYPE
```

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nQueryIdx</td>
<td>DINT</td>
<td>Query descriptor index</td>
</tr>
<tr>
<td>nTrainIdx</td>
<td>DINT</td>
<td>Train descriptor index</td>
</tr>
<tr>
<td>nImageIdx</td>
<td>DINT</td>
<td>Train image index</td>
</tr>
<tr>
<td>fDistance</td>
<td>REAL</td>
<td>Distance between the descriptors (smaller distance means better match)</td>
</tr>
</tbody>
</table>

#### 6.2.5.7 TcVnImageInfo

Shows image information.

**Syntax**

**Definition:**

```c
TYPE TcVnImageInfo :
STRUCT
    nImageSize : ULINT;
    nWidth     : UDINT;
    nHeight    : UDINT;
    nXPadding  : UINT;
    nYPadding  : UINT;
    stPixelFormat : TcVnPixelFormat;
END_STRUCT
END_TYPE
```

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nImageSize</td>
<td>ULINT</td>
<td></td>
</tr>
<tr>
<td>nWidth</td>
<td>UDINT</td>
<td></td>
</tr>
<tr>
<td>nHeight</td>
<td>UDINT</td>
<td></td>
</tr>
<tr>
<td>nXPadding</td>
<td>UINT</td>
<td></td>
</tr>
<tr>
<td>nYPadding</td>
<td>UINT</td>
<td></td>
</tr>
<tr>
<td>stPixelFormat</td>
<td>TcVnPixelFormat;</td>
<td></td>
</tr>
</tbody>
</table>
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nImageSize</td>
<td>ULINT</td>
<td>Image size (number of pixels)</td>
</tr>
<tr>
<td>nWidth</td>
<td>UDINT</td>
<td>Image width</td>
</tr>
<tr>
<td>nHeight</td>
<td>UDINT</td>
<td>Image height</td>
</tr>
<tr>
<td>nXPadding</td>
<td>UINT</td>
<td>Image x-padding</td>
</tr>
<tr>
<td>nYPadding</td>
<td>UINT</td>
<td>Image y-padding</td>
</tr>
<tr>
<td>stPixelFormat</td>
<td>TcVnPixelFormat</td>
<td>Pixel format</td>
</tr>
</tbody>
</table>

6.2.5.8   TcVnKeyPoint

Describes a key point.

Syntax

Definition:

```c
TYPE TcVnKeyPoint :
STRUCT
    aPoint : TcVnPoint2_REAL;
    fDiameter : REAL;
    fAngle : REAL;
    fResponse : REAL;
    nOctave : DINT;
    nClassId : DINT;
END_STRUCT
END_TYPE
```

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aPoint</td>
<td>TcVnPoint2_REAL</td>
<td>Position</td>
</tr>
<tr>
<td>fDiameter</td>
<td>REAL</td>
<td>Diameter</td>
</tr>
<tr>
<td>fAngle</td>
<td>REAL</td>
<td>Angle</td>
</tr>
<tr>
<td>fResponse</td>
<td>REAL</td>
<td>Response</td>
</tr>
<tr>
<td>nOctave</td>
<td>DINT</td>
<td>Octave</td>
</tr>
<tr>
<td>nClassId</td>
<td>DINT</td>
<td>Class ID</td>
</tr>
</tbody>
</table>

6.2.5.9   TcVnMatrix

Offers a user-defined matrix with variable rows, columns and element-type.

Syntax

Definition:

```c
TYPE TcVnMatrix :
STRUCT
    nRows : UDINT;
    nCols : UDINT;
    eType : ETcVnElementType;
    pData : PVOID;
END_STRUCT
END_TYPE
```
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nRows</td>
<td>UDINT</td>
<td>Number of rows</td>
</tr>
<tr>
<td>nCols</td>
<td>UDINT</td>
<td>Number of columns</td>
</tr>
<tr>
<td>eType</td>
<td>ETcVnElementType</td>
<td>Element type</td>
</tr>
<tr>
<td>pData</td>
<td>PVOID</td>
<td>Pointer to the data</td>
</tr>
</tbody>
</table>

#### 6.2.5.10 TcVnMoments

Offers image or contour moments.

**Syntax**

**Definition:**

```c
TYPE TcVnMoments :
  STRUCT
    fM00  : LREAL;
    fM10  : LREAL;
    fM01  : LREAL;
    fM20  : LREAL;
    fM11  : LREAL;
    fM02  : LREAL;
    fM21  : LREAL;
    fM12  : LREAL;
    fM03  : LREAL;
    fMu20 : LREAL;
    fMu11 : LREAL;
    fMu02 : LREAL;
    fMu30 : LREAL;
    fMu21 : LREAL;
    fMu12 : LREAL;
    fMu03 : LREAL;
    fNu20 : LREAL;
    fNu11 : LREAL;
    fNu02 : LREAL;
    fNu30 : LREAL;
    fNu21 : LREAL;
    fNu12 : LREAL;
    fNu03 : LREAL;
  END_STRUCT
END_TYPE
```
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fM00</td>
<td>LREAL</td>
<td>Spatial moment 00</td>
</tr>
<tr>
<td>fM10</td>
<td>LREAL</td>
<td>Spatial moment 10</td>
</tr>
<tr>
<td>fM01</td>
<td>LREAL</td>
<td>Spatial moment 01</td>
</tr>
<tr>
<td>fM20</td>
<td>LREAL</td>
<td>Spatial moment 20</td>
</tr>
<tr>
<td>fM11</td>
<td>LREAL</td>
<td>Spatial moment 11</td>
</tr>
<tr>
<td>fM02</td>
<td>LREAL</td>
<td>Spatial moment 02</td>
</tr>
<tr>
<td>fM30</td>
<td>LREAL</td>
<td>Spatial moment 30</td>
</tr>
<tr>
<td>fM21</td>
<td>LREAL</td>
<td>Spatial moment 21</td>
</tr>
<tr>
<td>fM12</td>
<td>LREAL</td>
<td>Spatial moment 12</td>
</tr>
<tr>
<td>fM03</td>
<td>LREAL</td>
<td>Spatial moment 03</td>
</tr>
<tr>
<td>fMu20</td>
<td>LREAL</td>
<td>Central moment 20</td>
</tr>
<tr>
<td>fMu11</td>
<td>LREAL</td>
<td>Central moment 11</td>
</tr>
<tr>
<td>fMu02</td>
<td>LREAL</td>
<td>Central moment 02</td>
</tr>
<tr>
<td>fMu30</td>
<td>LREAL</td>
<td>Central moment 30</td>
</tr>
<tr>
<td>fMu21</td>
<td>LREAL</td>
<td>Central moment 21</td>
</tr>
<tr>
<td>fMu12</td>
<td>LREAL</td>
<td>Central moment 12</td>
</tr>
<tr>
<td>fMu03</td>
<td>LREAL</td>
<td>Central moment 03</td>
</tr>
<tr>
<td>fNu20</td>
<td>LREAL</td>
<td>Normalized central moment 20</td>
</tr>
<tr>
<td>fNu11</td>
<td>LREAL</td>
<td>Normalized central moment 11</td>
</tr>
<tr>
<td>fNu02</td>
<td>LREAL</td>
<td>Normalized central moment 02</td>
</tr>
<tr>
<td>fNu30</td>
<td>LREAL</td>
<td>Normalized central moment 30</td>
</tr>
<tr>
<td>fNu21</td>
<td>LREAL</td>
<td>Normalized central moment 21</td>
</tr>
<tr>
<td>fNu12</td>
<td>LREAL</td>
<td>Normalized central moment 12</td>
</tr>
<tr>
<td>fNu03</td>
<td>LREAL</td>
<td>Normalized central moment 03</td>
</tr>
</tbody>
</table>

6.2.5.11 TcVnParamsAGAST

Offers parameters for AGAST method.

Syntax

Definition:

```plaintext
TYPE TcVnParamsAGAST :
STRUCT
    nThreshold    : DINT;
    bNonMaxSuppression : BOOL;
    eType         : ETcVnKeypointDetectionTypeAGAST;
END_STRUCT
END_TYPE
```
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nThreshold</td>
<td>DINT</td>
<td>10</td>
<td>Threshold for the intensity difference between the center pixel and its surrounding circle</td>
</tr>
<tr>
<td>bNonMaxSuppression</td>
<td>BOOL</td>
<td>TRUE</td>
<td>If true, non-maximum suppression is applied.</td>
</tr>
<tr>
<td>eType</td>
<td>ETCvNKeypointDetectionTypeAGAST</td>
<td>TCVN_KDT_AGAST_9_16</td>
<td>Neighborhood type</td>
</tr>
</tbody>
</table>

### 6.2.5.12 TcVnParamsAKAZE

Offers parameters for AKAZE method.

#### Syntax

**Definition:**

```plaintext
TYPE TcVnParamsAKAZE :
STRUCT
    eDescrType        : ETCvNFeatureDescriptorTypeAKAZE;
    nDescrSize        : UDINT;
    nDescrChannels    : UDINT;
    fThreshold        : REAL;
    nOctaves          : UDINT;
    nOctaveLayers     : UDINT;
    eDiffusivity      : ETCvNDiffusivityTypeKAZE;
END_STRUCT
END_TYPE
```

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eDescrType</td>
<td>ETCvNFeatureDescriptorTypeAKAZE</td>
<td>TCVN_FDT_AKAZE_MLB</td>
<td>Type of the descriptor</td>
</tr>
<tr>
<td>nDescrSize</td>
<td>UDINT</td>
<td>0</td>
<td>Size of the descriptor in bits (only for MLDB; 0 = full size)</td>
</tr>
<tr>
<td>nDescrChannels</td>
<td>UDINT</td>
<td>3</td>
<td>Number of descriptor channels (currently only 3 supported for MLDB)</td>
</tr>
<tr>
<td>fThreshold</td>
<td>REAL</td>
<td>0.001</td>
<td>Detector response threshold</td>
</tr>
<tr>
<td>nOctaves</td>
<td>UDINT</td>
<td>2</td>
<td>Maximum octave evolution</td>
</tr>
<tr>
<td>nOctaveLayers</td>
<td>UDINT</td>
<td>1</td>
<td>Number of sublevels per scale level</td>
</tr>
<tr>
<td>eDiffusivity</td>
<td>ETCvNDiffusivityTypeKAZE</td>
<td>TCVN_DT1_KAZE_PM_G_2</td>
<td>Diffusivity type</td>
</tr>
</tbody>
</table>

### 6.2.5.13 TcVnParamsBlobDetection

Offers parameters for blob detection.
Definition:

```c
TYPE TcVnParamsBlobDetection :
  STRUCT
    bFilterByArea           : BOOL;
    bFilterByCircularity   : BOOL;
    bFilterByConvexity     : BOOL;
    bFilterByEccentricity  : BOOL;
    bFilterByInertiaRatio  : BOOL;
    fMinArea               : REAL;
    fMaxArea               : REAL;
    fMinCircularity        : REAL;
    fMaxCircularity        : REAL;
    fMinConvexity          : REAL;
    fMaxConvexity          : REAL;
    fMinEccentricity       : REAL;
    fMaxEccentricity       : REAL;
    fMinInertiaRatio       : REAL;
    fMaxInertiaRatio       : REAL;
    eThresholdType         : ETcVnThresholdType;
    fMinThreshold          : REAL;
    fMaxThreshold          : REAL;
    fThresholdStep         : REAL;
    fMinBlobDistance       : REAL;
    nMinRepeatability      : UDINT;
    eBlobCombination       : ETcVnBlobCombination;
  END_STRUCT
END_TYPE
```
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bFilterByArea</td>
<td>BOOL</td>
<td>TRUE</td>
<td>Enable filtering by area (fMinArea &lt;= area &lt;= fMaxArea); strongly recommended to activate for filtering noise with fMinArea.</td>
</tr>
<tr>
<td>bFilterByCircularity</td>
<td>BOOL</td>
<td>FALSE</td>
<td>Enable filtering by circularity (fMinCircularity &lt;= circularity(4pi * area / perimeter^2) &lt;= fMaxCircularity).</td>
</tr>
<tr>
<td>bFilterByConvexity</td>
<td>BOOL</td>
<td>FALSE</td>
<td>Enable filtering by convexity (fMinConvexity &lt;= convexity(area / hullArea) &lt;= fMaxConvexity).</td>
</tr>
<tr>
<td>bFilterByEccentricity</td>
<td>BOOL</td>
<td>FALSE</td>
<td>Enable filtering by eccentricity (fMinEccentricity &lt;= eccentricity &lt;= fMaxEccentricity).</td>
</tr>
<tr>
<td>bFilterByInertiaRatio</td>
<td>BOOL</td>
<td>FALSE</td>
<td>Enable filtering by inertia ratio (fMinInertiaRatio &lt;= inertia ratio &lt;= fMaxInertiaRatio).</td>
</tr>
<tr>
<td>fMinArea</td>
<td>REAL</td>
<td>10</td>
<td>Min estimated blob area in pixel</td>
</tr>
<tr>
<td>fMaxArea</td>
<td>REAL</td>
<td>100000000</td>
<td>Max estimated blob area in pixel</td>
</tr>
<tr>
<td>fMinCircularity</td>
<td>REAL</td>
<td>0</td>
<td>Min circularity (1.0: ideal circle, &lt; 1: less circular, 0: not circular at all)</td>
</tr>
<tr>
<td>fMaxCircularity</td>
<td>REAL</td>
<td>1</td>
<td>Max circularity (1.0: ideal circle, &lt; 1: less circular, 0: not circular at all)</td>
</tr>
<tr>
<td>fMinConvexity</td>
<td>REAL</td>
<td>0</td>
<td>Min convexity (1.0: blob fully convex, &lt; 1: less convex)</td>
</tr>
<tr>
<td>fMaxConvexity</td>
<td>REAL</td>
<td>1</td>
<td>Max convexity (1.0: blob fully convex, &lt; 1: less convex)</td>
</tr>
<tr>
<td>fMinEccentricity</td>
<td>REAL</td>
<td>0</td>
<td>Min eccentricity (0.0: circular, 1.0: linear)</td>
</tr>
<tr>
<td>fMaxEccentricity</td>
<td>REAL</td>
<td>1</td>
<td>Max eccentricity (0.0: circular, 1.0: linear)</td>
</tr>
<tr>
<td>fMinInertiaRatio</td>
<td>REAL</td>
<td>0</td>
<td>Min inertia ratio (1.0: equal width and height, 0.0: linear)</td>
</tr>
<tr>
<td>fMaxInertiaRatio</td>
<td>REAL</td>
<td>1</td>
<td>Max inertia ratio (1.0: equal width and height, 0.0: linear)</td>
</tr>
<tr>
<td>eThresholdType</td>
<td>T CvThresholdType [1] 201</td>
<td>TCVN_TT_BINARY</td>
<td>Threshold type for internally applied threshold method (OTSU_XXX only supported for USINT images).</td>
</tr>
</tbody>
</table>
## 6.2.5.14 TcVnParamsBRISK

Offers parameters for BRISK method.

**Syntax**

**Definition:**

```plaintext
TYPE TcVnParamsBRISK :
  STRUCT
    nThreshold : DINT;
    nOctaves : UDINT;
    fPatternScale : REAL;
  END_STRUCT
END_TYPE
```

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nThreshold</td>
<td>DINT</td>
<td>30</td>
<td>Detection threshold</td>
</tr>
<tr>
<td>nOctaves</td>
<td>UDINT</td>
<td>3</td>
<td>Detection octaves (0 for single scale)</td>
</tr>
<tr>
<td>fPatternScale</td>
<td>REAL</td>
<td>1</td>
<td>Scale factor for the neighborhood pattern</td>
</tr>
</tbody>
</table>
6.2.5.15  TcVnParamsFAST

Offers parameters for FAST method.

**Syntax**

**Definition:**

```plaintext
TYPE TcVnParamsFAST :
  STRUCT
    nThreshold       : DINT;
    bNonMaxSuppression : BOOL;
    eType            : ETcVnKeypointDetectionTypeFAST;
  END_STRUCT
END_TYPE
```

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nThreshold</td>
<td>DINT</td>
<td>10</td>
<td>Threshold for the intensity difference between the center pixel and its surrounding circle</td>
</tr>
<tr>
<td>bNonMaxSuppression</td>
<td>BOOL</td>
<td>TRUE</td>
<td>If true, non-maximum suppression is applied.</td>
</tr>
<tr>
<td>eType</td>
<td>ETcVnKeypointDetectionTypeFAST</td>
<td>TCVN_KDT_FAST_9_16</td>
<td>Neighborhood type</td>
</tr>
</tbody>
</table>

6.2.5.16  TcVnParamsGFTT

Offers parameters for GFTT method.

**Syntax**

**Definition:**

```plaintext
TYPE TcVnParamsGFTT :
  STRUCT
    nMaxCorners       : UDINT;
    fQualityLevel      : LREAL;
    fMinDistance     : LREAL;
    nBlockSize        : UDINT;
    bUseHarrisDetector: BOOL;
    fHarrisK          : LREAL;
  END_STRUCT
END_TYPE
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nMaxCorners</td>
<td>UDINT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fQualityLevel</td>
<td>LREAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fMinDistance</td>
<td>LREAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nBlockSize</td>
<td>UDINT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bUseHarrisDetector</td>
<td>BOOL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fHarrisK</td>
<td>LREAL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nMaxCorners</td>
<td>UDINT</td>
<td>1000</td>
<td>Maximum number of corners to return (strongest ones)</td>
</tr>
<tr>
<td>fQualityLevel</td>
<td>LREAL</td>
<td>0.01</td>
<td>Minimum accepted corner quality, relative to the strongest one</td>
</tr>
<tr>
<td>fMinDistance</td>
<td>LREAL</td>
<td>1</td>
<td>Minimum euclidean distance between returned keypoints</td>
</tr>
<tr>
<td>nBlockSize</td>
<td>UDINT</td>
<td>3</td>
<td>Neighborhood block size</td>
</tr>
<tr>
<td>bUseHarrisDetector</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If true, a Harris detector is used instead of the default method.</td>
</tr>
<tr>
<td>fHarrisK</td>
<td>LREAL</td>
<td>0.04</td>
<td>Free parameter of the Harris detector</td>
</tr>
</tbody>
</table>

6.2.5.17 TcVnParamsKAZE

Offers parameters for KAZE method

Syntax

Definition:

```cpp
TYPE TcVnParamsKAZE :
STRUCT
  bExtended     : BOOL;
  bUpright      : BOOL;
  fThreshold    : REAL;
  nOctaves      : UDINT;
  nOctaveLayers : UDINT;
  eDiffusivity  : ETcVnDiffusivityTypeKAZE;
END_STRUCT
END_TYPE
```

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bExtended</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If true, the descriptor size is extended from the default 64 byte to 128 byte.</td>
</tr>
<tr>
<td>bUpright</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If true, upright (rotation dependent) descriptors are computed.</td>
</tr>
<tr>
<td>fThreshold</td>
<td>REAL</td>
<td>0.001</td>
<td>Detector response threshold</td>
</tr>
<tr>
<td>nOctaves</td>
<td>UDINT</td>
<td>4</td>
<td>Maximum octave evolution of the image</td>
</tr>
<tr>
<td>nOctaveLayers</td>
<td>UDINT</td>
<td>2</td>
<td>Number of sublevels per scale level</td>
</tr>
<tr>
<td>eDiffusivity</td>
<td>ETcVnDiffusivityTypeKAZE</td>
<td>TCVN_DT1_KAZE_PM_G2</td>
<td>Diffusivity type</td>
</tr>
</tbody>
</table>
### 6.2.5.18 TcVnParamsMSER

Offers parameters for MSER method

#### Syntax

Definition:

```plaintext
TYPE TcVnParamsMSER :
  STRUCT
    nDelta : DINT;
    nMinArea : UDINT;
    nMaxArea : UDINT;
    fMaxVariation : LREAL;
    fMinDiversity : LREAL;
    nMaxEvolution : UDINT;
    fAreaThreshold : LREAL;
    fMinMargin : LREAL;
    nEdgeBlurSize : UDINT;
  END_STRUCT
END_TYPE
```

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nDelta</td>
<td>DINT</td>
<td>5</td>
<td>Delta for size comparison</td>
</tr>
<tr>
<td>nMinArea</td>
<td>UDINT</td>
<td>60</td>
<td>Minimum blob size</td>
</tr>
<tr>
<td>nMaxArea</td>
<td>UDINT</td>
<td>14400</td>
<td>Maximum blob size</td>
</tr>
<tr>
<td>fMaxVariation</td>
<td>LREAL</td>
<td>0.25</td>
<td>Maximum blob size variation</td>
</tr>
<tr>
<td>fMinDiversity</td>
<td>LREAL</td>
<td>0.2</td>
<td>Minimum MSER diversity</td>
</tr>
<tr>
<td>nMaxEvolution</td>
<td>UDINT</td>
<td>200</td>
<td>Maximum evolution steps (only used for color images)</td>
</tr>
<tr>
<td>fAreaThreshold</td>
<td>LREAL</td>
<td>1.01</td>
<td>Area threshold to cause re-initialization (only used for color images)</td>
</tr>
<tr>
<td>fMinMargin</td>
<td>LREAL</td>
<td>0.003</td>
<td>Minimum margin (only used for color images)</td>
</tr>
<tr>
<td>nEdgeBlurSize</td>
<td>UDINT</td>
<td>5</td>
<td>Aperture size for edge bluring (only used for color images)</td>
</tr>
</tbody>
</table>

### 6.2.5.19 TcVnParamsORB

Offers parameters for ORB method.

#### Syntax

Definition:

```plaintext
TYPE TcVnParamsORB :
  STRUCT
    nMaxPoints : UDINT;
    fPyramidScale : REAL;
    nPyramidLevels : UDINT;
    nEdgeThreshold : UDINT;
    nFirstLevel : UDINT;
    nBriefPoints : UDINT;
    eScoreType : ETcVnFeatureScoreTypeORB;
    nPatchSize : UDINT;
    nFastThreshold : DINT;
  END_STRUCT
END_TYPE
```
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nMaxPoints</td>
<td>UDINT</td>
<td>500</td>
<td>Maximum number of returned keypoints</td>
</tr>
<tr>
<td>fPyramidScale</td>
<td>REAL</td>
<td>1.2</td>
<td>Pyramid decimation ratio (must be greater than 1, should be smaller than 2)</td>
</tr>
<tr>
<td>nPyramidLevels</td>
<td>UDINT</td>
<td>8</td>
<td>Number of pyramid levels</td>
</tr>
<tr>
<td>nEdgeThreshold</td>
<td>UDINT</td>
<td>31</td>
<td>Size of the border, where no features are detected (should match nPatchSize)</td>
</tr>
<tr>
<td>nFirstLevel</td>
<td>UDINT</td>
<td>0</td>
<td>First pyramid level (currently, only 0 is supported)</td>
</tr>
<tr>
<td>nBriefPoints</td>
<td>UDINT</td>
<td>2</td>
<td>Number of points to produce each BRIEF descriptor element (2, 3, 4)</td>
</tr>
<tr>
<td>eScoreType</td>
<td>ETcVnFeatureScoreTypeORB</td>
<td>TCVN_FST_ORB_HARRIS</td>
<td>Score type (HARRIS is more stable but slightly slower than FAST)</td>
</tr>
<tr>
<td>nPatchSize</td>
<td>UDINT</td>
<td>31</td>
<td>Patch size of the BRIEF descriptor</td>
</tr>
<tr>
<td>nFastThreshold</td>
<td>DINT</td>
<td>20</td>
<td>Threshold for the FAST keypoint detection</td>
</tr>
</tbody>
</table>

### 6.2.5.20 TcVnParamsSB

Offers parameters for SB method (a simple blob detector with multiple thresholds).

#### Syntax

**Definition:**

```c
TYPE TcVnParamsSB :
STRUCT
  bFilterByArea : BOOL;
  bFilterByCircularity : BOOL;
  bFilterByColor : BOOL;
  bFilterByConvexity : BOOL;
  bFilterByInertia : BOOL;
  fMinArea : REAL;
  fMaxArea : REAL;
  fMinCircularity : REAL;
  fMaxCircularity : REAL;
  nBlobColor : USINT;
  fMinConvexity : REAL;
  fMaxConvexity : REAL;
  fMinInertiaRatio : REAL;
  fMaxInertiaRatio : REAL;
  fMinBlobDist : REAL;
  fMinRepeatability : UDINT;
  fMinThreshold : REAL;
  fMaxThreshold : REAL;
  fThresholdStep : REAL;
END_STRUCT
END_TYPE
```
## Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bFilterByArea</code></td>
<td>BOOL</td>
<td>TRUE</td>
<td>Enable filtering by area (fMinArea &lt;= area &lt; fMaxArea).</td>
</tr>
<tr>
<td><code>bFilterByCircularity</code></td>
<td>BOOL</td>
<td>FALSE</td>
<td>Enable filtering by circularity (fMinCircularity &lt;= circularity(4pi * area / perimeter^2) &lt; fMaxCircularity).</td>
</tr>
<tr>
<td><code>bFilterByColor</code></td>
<td>BOOL</td>
<td>FALSE</td>
<td>Enable filtering by color (thresholdedColor(0 or 255) = nBlobColor).</td>
</tr>
<tr>
<td><code>bFilterByConvexity</code></td>
<td>BOOL</td>
<td>FALSE</td>
<td>Enable filtering by convexity (fMinConvexity &lt;= convexity(area / hullArea) &lt; fMaxConvexity).</td>
</tr>
<tr>
<td><code>bFilterByInertia</code></td>
<td>BOOL</td>
<td>FALSE</td>
<td>Enable filtering by inertia ratio (fMinInertiaRatio &lt;= inertia ratio &lt; fMaxInertiaRatio).</td>
</tr>
<tr>
<td><code>fMinArea</code></td>
<td>REAL</td>
<td>25</td>
<td>Min estimated blob area in pixel</td>
</tr>
<tr>
<td><code>fMaxArea</code></td>
<td>REAL</td>
<td>15000</td>
<td>Max estimated blob area in pixel</td>
</tr>
<tr>
<td><code>fMinCircularity</code></td>
<td>REAL</td>
<td>0</td>
<td>Min circularity (1.0: ideal circle, &lt; 1: less circular, 0: not circular at all)</td>
</tr>
<tr>
<td><code>fMaxCircularity</code></td>
<td>REAL</td>
<td>1</td>
<td>Max circularity (1.0: ideal circle, &lt; 1: less circular, 0: not circular at all)</td>
</tr>
<tr>
<td><code>nBlobColor</code></td>
<td>USINT</td>
<td>255</td>
<td>0 or 255</td>
</tr>
<tr>
<td><code>fMinConvexity</code></td>
<td>REAL</td>
<td>0</td>
<td>Min convexity (1.0: blob fully convex, &lt; 1: less convex)</td>
</tr>
<tr>
<td><code>fMaxConvexity</code></td>
<td>REAL</td>
<td>1</td>
<td>Max convexity (1.0: blob fully convex, &lt; 1: less convex)</td>
</tr>
<tr>
<td><code>fMinInertiaRatio</code></td>
<td>REAL</td>
<td>0</td>
<td>Min inertia ratio (0.0 .. 1.0)</td>
</tr>
<tr>
<td><code>fMaxInertiaRatio</code></td>
<td>REAL</td>
<td>1</td>
<td>Max inertia ratio (0.0 .. 1.0)</td>
</tr>
<tr>
<td><code>fMinBlobDist</code></td>
<td>REAL</td>
<td>5</td>
<td>Min distance between different blobs</td>
</tr>
<tr>
<td><code>nMinRepeatability</code></td>
<td>UDINT</td>
<td>2</td>
<td>Min number of same detected blobs (dist &lt; fMinBlobDist) in different thresholds</td>
</tr>
<tr>
<td><code>fMinThreshold</code></td>
<td>REAL</td>
<td>30</td>
<td>Min threshold (start)</td>
</tr>
<tr>
<td><code>fMaxThreshold</code></td>
<td>REAL</td>
<td>225</td>
<td>Max threshold (stop)</td>
</tr>
<tr>
<td><code>fThresholdStep</code></td>
<td>REAL</td>
<td>10</td>
<td>Threshold step between min and max threshold</td>
</tr>
</tbody>
</table>
6.2.5.21 TcVnPixelFormat
Contains detailed information about the pixel format.

Syntax

Definition:

TYPE TcVnPixelFormat :
  STRUCT
    bSupported : BIT;
    bSigned   : BIT;
    bPlanar   : BIT;
    bFloat    : BIT;
    nChannels : BYTE;
    ePixelEncoding : ETcVnPixelEncoding;
    ePixelPackMode : ETcVnPixelPackMode;
    nElementSize : UINT;
    nTotalSize  : UINT;
  END_STRUCT
END_TYPE

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bSupported</td>
<td>BIT</td>
<td>If false, the pixel format is not supported.</td>
</tr>
<tr>
<td>bSigned</td>
<td>BIT</td>
<td>If true, pixel intensities are signed values.</td>
</tr>
<tr>
<td>bPlanar</td>
<td>BIT</td>
<td>If true, the image channels are stored planar instead of interleaved (e.g. RRRRGGGGGBBBBB instead of RGBRGBRGBRGBRGB).</td>
</tr>
<tr>
<td>bFloat</td>
<td>BIT</td>
<td>If true, the pixel format is floating point.</td>
</tr>
<tr>
<td>nChannels</td>
<td>BYTE</td>
<td>Number of channels</td>
</tr>
<tr>
<td>ePixelEncoding</td>
<td>ETcVnPixelEncoding</td>
<td>Pixel encoding</td>
</tr>
<tr>
<td>ePixelPackMode</td>
<td>ETcVnPixelPackMode</td>
<td>Pixel pack mode</td>
</tr>
<tr>
<td>nElementSize</td>
<td>UINT</td>
<td>Size (bit) of a single pixel channel</td>
</tr>
<tr>
<td>nTotalSize</td>
<td>UINT</td>
<td>Size (bit) of all pixel channels</td>
</tr>
</tbody>
</table>

6.2.5.22 TcVnRectangle_DINT
Contains origin and size of a rectangle.

Syntax

Definition:

TYPE TcVnRectangle_DINT :
  STRUCT
    nX    : DINT;
    nY    : DINT;
    nWidth: DINT;
    nHeight: DINT;
  END_STRUCT
END_TYPE
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nX</td>
<td>DINT</td>
<td>X coordinate of the top-left corner</td>
</tr>
<tr>
<td>nY</td>
<td>DINT</td>
<td>Y coordinate of the top-left corner</td>
</tr>
<tr>
<td>nWidth</td>
<td>DINT</td>
<td>Width</td>
</tr>
<tr>
<td>nHeight</td>
<td>DINT</td>
<td>Height</td>
</tr>
</tbody>
</table>

#### 6.2.5.23 TcVnRectangle_UDINT

Contains origin and size of a rectangle.

**Syntax**

**Definition:**

```plaintext
TYPE TcVnRectangle_UDINT :
  STRUCT
    nX  : UDINT;
    nY  : UDINT;
    nWidth : UDINT;
    nHeight : UDINT;
  END_STRUCT
END_TYPE
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nX</td>
<td>UDINT</td>
<td>X coordinate of the top-left corner</td>
</tr>
<tr>
<td>nY</td>
<td>UDINT</td>
<td>Y coordinate of the top-left corner</td>
</tr>
<tr>
<td>nWidth</td>
<td>UDINT</td>
<td>Width</td>
</tr>
<tr>
<td>nHeight</td>
<td>UDINT</td>
<td>Height</td>
</tr>
</tbody>
</table>

#### 6.2.5.24 TcVnRotatedRectangle

Contains center, size and angle of a rotated rectangle.

**Syntax**

**Definition:**

```plaintext
TYPE TcVnRotatedRectangle :
  STRUCT
    aCenter : TcVnPoint2_REAL;
    stSize  : TcVnSize2_REAL;
    fAngle  : REAL;
  END_STRUCT
END_TYPE
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aCenter</td>
<td>TcVnPoint2_REAL [151]</td>
<td>Center point</td>
</tr>
<tr>
<td>stSize</td>
<td>TcVnSize2_REAL [223]</td>
<td>Size composed of fWidth and fHeight</td>
</tr>
<tr>
<td>fAngle</td>
<td>REAL</td>
<td>Angle in degree</td>
</tr>
</tbody>
</table>
6.2.5.25 TcVnSize2_REAL

Contains width and height.

Syntax

Definition:

```plaintext
TYPE TcVnSize2_REAL :
  STRUCT
    fWidth  : REAL;
    fHeight : REAL;
  END_STRUCT
END_TYPE
```

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fWidth</td>
<td>REAL</td>
<td>Width</td>
</tr>
<tr>
<td>fHeight</td>
<td>REAL</td>
<td>Height</td>
</tr>
</tbody>
</table>

6.3 Interfaces

Interfaces do not require a license.

Grouping of interfaces

All interfaces of the TwinCAT Vision library are sorted thematically into the following groups:

- Images
- Container
- Machine Learning
- Miscellaneous

6.3.1 Containers

This group contains interfaces for handling containers.

General information

A container is basically a vector of several elements. The number of elements can be changed dynamically. The corresponding memory in the router memory is automatically allocated and released for this purpose.

Container types

All elements of a container have the same type. Containers are characterized by a unique GUID based on the type of elements they contain. For ease of use, these GUIDs are defined by constants. All type designations for containers begin with "ContainerType_Vector_" (with the exception of "ContainerType_String_SINT"). Type requirements are documented for all API elements that make use of containers.

The container type can be determined as follows:

```plaintext
ipContainer.GetTypeElementGuid(nTypeGuid);
```
6.3.1.1 Function Blocks
This group contains interfaces for the implementation of certain function blocks.

6.3.1.1.1 ITcVnCustomContainerOperation_ITcVnContainer
Offers an interface for custom, elementwise container operations (for containers of containers) with up to 3 different containers.

Inheritance Hierarchy
ITcVnCustomContainerOperation_ITcVnContainer

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>ITcVnCustomContainerOperation_ITcVnContainer</td>
<td>Executes the custom operation on the container elements.</td>
</tr>
</tbody>
</table>

Related functions
- F_VN_CustomElementWiseContainerOperation_ITcVnContainer [864]

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

Execute

```
Execute
  ipElement1 : ITcVnContainer
  ipElement2 : ITcVnContainer
  ipElement3 : ITcVnContainer
  HRESULT    Execute
```

Executes the custom operation on the container elements.

Syntax

```
METHOD Execute : HRESULT
VAR_INPUT
  ipElement1 : ITcVnContainer;
  ipElement2 : ITcVnContainer;
  ipElement3 : ITcVnContainer;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipElement1</td>
<td>ITcVnContainer [345]</td>
<td>Current element of the 1st container.</td>
</tr>
<tr>
<td>ipElement2</td>
<td>ITcVnContainer [345]</td>
<td>Current element of the 2nd container.</td>
</tr>
<tr>
<td>ipElement3</td>
<td>ITcVnContainer [345]</td>
<td>Current element of the 3rd container.</td>
</tr>
</tbody>
</table>
6.3.1.1.2  ITcVnCustomContainerOperation_ITcVnForwardIterator

Offers an interface for custom, elementwise container operations with up to 3 different containers.

Inheritance Hierarchy
ITcVnCustomContainerOperation_ITcVnForwardIterator

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute [225]</td>
<td>ITcVnCustomContainerOperation_ITcVnForwardIterator</td>
<td>Executes the custom operation on the container elements.</td>
</tr>
</tbody>
</table>

Related functions
•  F_VN_CustomElementWiseContainerOperation_ITcVnForwardIterator [865]

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

Execute

```
METHOD Execute : HRESULT
VARR_INPUT
  ipElement1 : ITcVnForwardIterator;
  ipElement2 : ITcVnForwardIterator;
  ipElement3 : ITcVnForwardIterator;
END_VAR

execute
```

Executes the custom operation on the container elements.

Syntax

Definition:

```
METHOD Execute : HRESULT
VARR_INPUT
  ipElement1 : ITcVnForwardIterator;
  ipElement2 : ITcVnForwardIterator;
  ipElement3 : ITcVnForwardIterator;
END_VAR
```
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipElement1</td>
<td>ITcVnForwardIterator</td>
<td>Current element of the 1st container.</td>
</tr>
<tr>
<td>ipElement2</td>
<td>ITcVnForwardIterator</td>
<td>Current element of the 2nd container.</td>
</tr>
<tr>
<td>ipElement3</td>
<td>ITcVnForwardIterator</td>
<td>Current element of the 3rd container.</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

6.3.1.1.3 ITcVnCustomElementCondition_ITcVnContainer

Offers an interface for a custom condition computed for a container.

**Inheritance Hierarchy**

ITcVnCustomElementCondition_ITcVnContainer

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Condition</td>
<td>ITcVnCustomElementCondition_ITcVnContainer</td>
<td>Evaluates the condition for the container.</td>
</tr>
</tbody>
</table>

**Related functions**

- F_VN_CopyContainerElementsConditional_ITcVnContainer [690]

**Samples**

- Selections of container elements [1364]

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**Condition**

```c
ipElement ITcVnContainer BOOL Condition
```

Evaluates the condition for the container.

**Syntax**

**Definition:**

---

Version: 1.3
METHOD Condition : BOOL
VAR_INPUT
  ipElement : ITcVnContainer;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipElement</td>
<td>ITcVnContainer [345]</td>
<td>The container for which the condition is evaluated.</td>
</tr>
</tbody>
</table>

Return value

BOOL

6.3.1.1.4 ITcVnCustomElementCondition_ITcVnForwardIterator

Offers an interface for a custom condition computed for an element represented by an ITcVnForwardIterator.

Inheritance Hierarchy

ITcVnCustomElementCondition_ITcVnForwardIterator

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Condition</td>
<td>ITcVnCustomElementCondition_ITcVnForwardIterator</td>
<td>Evaluates the condition for the provided element.</td>
</tr>
</tbody>
</table>

Related functions

- F_VN_CopyContainerElementsConditional_ITcVnForwardIterator [691]

Samples

- Selections of container elements [1364]

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

Condition

Evaluates the condition for the provided element.

Syntax

Definition:

METHOD Condition : BOOL
VAR_INPUT
  ipElement : ITcVnForwardIterator;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipElement</td>
<td>ITcVnForwardIterator</td>
<td>Element for which the condition is evaluated.</td>
</tr>
</tbody>
</table>

## Return value

BOOL

### 6.3.1.2 ITcVnAccess

The interface ITcVnAccess is for accessing elements of a container and is provided by an iterator. Unlike ITcVnRandomAccess, only that element to which the associated iterator is pointing in that moment can be addressed.

#### 6.3.1.2.1 ITcVnAccess_DINT

Offers an access interface for DINT values.

### Inheritance Hierarchy

ITcUnknown | ITcVnAccess_DINT

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get</td>
<td>ITcVnAccess_DINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set</td>
<td>ITcVnAccess_DINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### Get

Get

**GetValue Reference To DINT** HRESULT Get

Gets the value

### Syntax

**Definition:**
METHOD Get : HRESULT
VAR_INPUT
  nValue : Reference To DINT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nValue</td>
<td>Reference To DINT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Set

Sets the value

Syntax

Definition:
METHOD Set : HRESULT
VAR_INPUT
  nValue : DINT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nValue</td>
<td>DINT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.3.1.2.2 ITcVnAccess_INT

Offers an access interface for INT values.

Inheritance Hierarchy

ITcUnknown [399]
ITcVnAccess_INT
### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [230]</td>
<td>ITcVnAccess_INT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [230]</td>
<td>ITcVnAccess_INT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### Get

**Syntax**

**Definition:**

```plaintext
METHOD Get : HRESULT
VAR_INPUT
  nValue : Reference To INT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nValue</td>
<td>Reference To INT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

### Set

**Syntax**

**Definition:**

```plaintext
METHOD Set : HRESULT
VAR_INPUT
  nValue : INT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nValue</td>
<td>INT</td>
<td></td>
</tr>
</tbody>
</table>
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nValue</td>
<td>INT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### 6.3.1.2.3 ITcVnAccess_LREAL

Offers an access interface for LREAL values.

#### Inheritance Hierarchy

ITcUnknown [399]

ITcVnAccess_LREAL

#### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITCUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITCUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITCUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [231]</td>
<td>ITcVnAccess_LREAL</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [232]</td>
<td>ITcVnAccess_LREAL</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

#### Get

```plaintext
Get

fValue  Reference To LREAL  HRESULT  Get
```

Gets the value

#### Syntax

**Definition:**

```plaintext
METHOD Get : HRESULT
VAR_INPUT
  fValue : Reference To LREAL;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fValue</td>
<td>Reference To LREAL</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Set

Sets the value

Syntax

Definition:

METHOD Set : HRESULT
VAR_INPUT
   fValue : LREAL;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fValue</td>
<td>LREAL</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.3.1.2.4 ITcVnAccess_REAL

Offers an access interface for REAL values.

Inheritance Hierarchy

ITcUnknown [399]

ITcVnAccess_REAL

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [233]</td>
<td>ITcVnAccess_REAL</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [233]</td>
<td>ITcVnAccess_REAL</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
Get

Get

fvValue  Reference To REAL  HRESULT  Get

Gets the value

Syntax

Definition:

METHOD Get : HRESULT
VAR_INPUT
  fvValue : Reference To REAL;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fvValue</td>
<td>Reference To REAL</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Set

Set

fvValue  REAL  HRESULT  Set

Sets the value

Syntax

Definition:

METHOD Set : HRESULT
VAR_INPUT
  fvValue : REAL;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fvValue</td>
<td>REAL</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.3.1.2.5  ITcVnAccess_SINT

Offers an access interface for SINT values.
Inheritance Hierarchy

ITcUnknown [399]
ITcVnAccess_SINT

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [234]</td>
<td>ITcVnAccess_SINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [234]</td>
<td>ITcVnAccess_SINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

Get

```
Get
  nValue Reference To SINT HRESULT Get
```

Gets the value

Syntax

Definition:

```c
METHOD Get : HRESULT
VAR_INPUT
  nValue : Reference To SINT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nValue</td>
<td>Reference To SINT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Set

```
Set
  nValue SINT HRESULT Set
```

Sets the value
Syntax

Definition:

<table>
<thead>
<tr>
<th>METHOD Set : HRESULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>VAR_INPUT</td>
</tr>
<tr>
<td>nValue : SINT;</td>
</tr>
<tr>
<td>END_VAR</td>
</tr>
</tbody>
</table>

## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nValue</td>
<td>SINT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

### 6.3.1.2.6 ITcVnAccess_TcVnDMatch

Offers an access interface for TcVnDMatch values.

**Inheritance Hierarchy**

ITcUnknown [399]

| ITcVnAccess_TcVnDMatch |

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [235]</td>
<td>ITcVnAccess_TcVnDMatch</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [236]</td>
<td>ITcVnAccess_TcVnDMatch</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## Get

Get stDMatch Reference To TcVnDMatch HRESULT Get

Gets the value

### Syntax

**Definition:**

[Image]
METHOD Get : HRESULT
VAR_INPUT
   stDMatch : Reference To TcVnDMatch;
END_VAR

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stDMatch</td>
<td>Reference To TcVnDMatch</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

**Inputs**

**Return value**

HRESULT [135]

**Set**

Sets the value

**Syntax**

**Definition:**

METHOD Set : HRESULT
VAR_INPUT
   stDMatch : Reference To TcVnDMatch;
END_VAR

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stDMatch</td>
<td>Reference To TcVnDMatch</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

**Inputs**

**Return value**

HRESULT [135]

6.3.1.2.7 ITcVnAccess_TcVnKeyPoint

Offers an access interface for TcVnKeyPoint values.

**Inheritance Hierarchy**

ITcUnknown [399]
   ITcVnAccess_TcVnKeyPoint
### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [237]</td>
<td>ITcVnAccess_TcVnKeyPoint</td>
<td>Gets the value.</td>
</tr>
<tr>
<td>Set [237]</td>
<td>ITcVnAccess_TcVnKeyPoint</td>
<td>Sets the value.</td>
</tr>
</tbody>
</table>

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### Get

**Get**

Input:

```
stKeyPoint Reference To TcVnKeyPoint  HRESULT Get
```

Gets the value.

**Syntax**

**Definition:**

```c
METHOD Get : HRESULT
VAR_INPUT
    stKeyPoint : Reference To TcVnKeyPoint;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stKeyPoint</td>
<td>Reference To TcVnKeyPoint [208]</td>
<td>Returns the value.</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

### Set

**Set**

Input:

```
setKeyPoint Reference To TcVnKeyPoint  HRESULT Set
```

Sets the value.

**Syntax**

**Definition:**

```c
METHOD Set : HRESULT
VAR_INPUT
    stKeyPoint : Reference To TcVnKeyPoint;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stKeyPoint</td>
<td>Reference To TcVnKeyPoint</td>
<td>The value to set.</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

### 6.3.1.2.8 ITCvNAccess_TcVnPoint2_DINT

Offers an access interface for TcVnPoint2_DINT values.

#### Inheritance Hierarchy

ITcUnknown [399]
- ITCvNAccess_TcVnPoint2_DINT

#### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITCUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITCUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITCUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [238]</td>
<td>ITCvNAccess_TcVnPoint2_DINT</td>
<td>Gets the value.</td>
</tr>
<tr>
<td>Set [239]</td>
<td>ITCvNAccess_TcVnPoint2_DINT</td>
<td>Sets the value.</td>
</tr>
</tbody>
</table>

#### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

#### Get

```plaintext
Get

aPoint : Reference To TcVnPoint2_DINT
HRESULT
Get```

Gets the value.

#### Syntax

**Definition:**

METHOD Get : HRESULT
VAR_INPUT
   aPoint : Reference To TcVnPoint2_DINT;
END_VAR
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aPoint</td>
<td>Reference To TcVnPoint2_DINT</td>
<td>Returns the value.</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Set

Sets the value.

Syntax

Definition:

METHOD Set : HRESULT
VAR_INPUT
    aPoint : Reference To TcVnPoint2_DINT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aPoint</td>
<td>Reference To TcVnPoint2_DINT</td>
<td>The value to set.</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.3.1.2.9 ITcVnAccess_TcVnPoint2_LREAL

Offers an access interface for TcVnPoint2_LREAL values.

Inheritance Hierarchy

ITcUnknown [399]
ITcVnAccess_TcVnPoint2_LREAL
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [240]</td>
<td>ITcVnAccess_TcVnPoint2_LREAL</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [240]</td>
<td>ITcVnAccess_TcVnPoint2_LREAL</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## Get

### Syntax

**Definition:**

```plaintext
METHOD Get : HRESULT
VAR_INPUT
  aPoint : Reference To TcVnPoint2_LREAL;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aPoint</td>
<td>Reference To TcVnPoint2_LREAL [151]</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

### Return value

**HRESULT [135]**

## Set

### Syntax

**Definition:**

```plaintext
METHOD Set : HRESULT
VAR_INPUT
  aPoint : Reference To TcVnPoint2_LREAL;
END_VAR
```
**API reference**

**TF7000 - TF7300**

**Version: 1.3**

### METHOD Set : HRESULT

**VAR_INPUT**

- `aPoint : Reference To TcVnPoint2_LREAL;`

**END_VAR**

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>aPoint</code></td>
<td><code>Reference To TcVnPoint2_LREAL</code></td>
<td>The value to set</td>
</tr>
</tbody>
</table>

#### Return value

**HRESULT**

**6.3.1.2.10 ITCvAccess_TcVnPoint2_REAL**

Offers an access interface for TcVnPoint2_REAL values.

#### Inheritance Hierarchy

- `ITcUnknown`  
- `ITCvAccess_TcVnPoint2_REAL`

#### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>TcAddRef</code></td>
<td><code>ITcUnknown</code></td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td><code>TcQueryInterface</code></td>
<td><code>ITcUnknown</code></td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td><code>TcRelease</code></td>
<td><code>ITcUnknown</code></td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td><code>Get</code></td>
<td><code>ITCvAccess_TcVnPoint2_REAL</code></td>
<td>Gets the value</td>
</tr>
<tr>
<td><code>Set</code></td>
<td><code>ITCvAccess_TcVnPoint2_REAL</code></td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

#### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

#### Get

```
METHOD Get : HRESULT
VAR_INPUT
  aPoint : Reference To TcVnPoint2_REAL;
END_VAR
```

Gets the value

#### Syntax

**Definition:**

```
METHOD Get : HRESULT
VAR_INPUT
  aPoint : Reference To TcVnPoint2_REAL;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aPoint</td>
<td>Reference To TcVnPoint2_REAL</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

### Set

Sets the value

#### Syntax

**Definition:**

METHOD Set : HRESULT  
VAR_INPUT
  aPoint : Reference To TcVnPoint2_REAL;
END_VAR

## Inheritance Hierarchy

ITcUnknown [399]  
ITcVnAccess_TcVnPoint3_LREAL

### 6.3.1.2.11 ITcVnAccess_TcVnPoint3_LREAL

Offers an access interface for TcVnPoint3_LREAL values.
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [243]</td>
<td>ITcVnAccess_TcVnPoint3 _LREAL</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [243]</td>
<td>ITcVnAccess_TcVnPoint3 _LREAL</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### Get

Get

```plaintext
Get

aPoint References To TcVnPoint3_LREAL HRESULT Get
```

Gets the value

### Syntax

**Definition:**

```plaintext
METHOD Get : HRESULT
VAR_INPUT
  aPoint : Reference To TcVnPoint3_LREAL;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aPoint</td>
<td>Reference To TcVnPoint3_LREAL [151]</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

### Set

Set

```plaintext
Set

aPoint References To TcVnPoint3_LREAL HRESULT Set
```

Sets the value

### Syntax

**Definition:**

```plaintext
METHOD Set : HRESULT
VAR_INPUT
  aPoint : Reference To TcVnPoint3_LREAL;
END_VAR
```
API reference

METHOD Set : HRESULT
VAR_INPUT
   aPoint : Reference To TcVnPoint3_LREAL;
END_VAR

Name | Type | Description
--- | --- | ---
aPoint | Reference To TcVnPoint3_REAL [151] | The value to set

**Return value**

HRESULT [135]

**6.3.1.2.12 ITcVnAccess_TcVnPoint3_REAL**

Offers an access interface for TcVnPoint3_REAL values.

**Inheritance Hierarchy**

ITcUnknown [399]
   ITcVnAccess_TcVnPoint3_REAL

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [244]</td>
<td>ITcVnAccess_TcVnPoint3_REAL</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [245]</td>
<td>ITcVnAccess_TcVnPoint3_REAL</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**Get**

Get

```
   aPoint Reference To TcVnPoint3_REAL HRESULT Get
```

Gets the value

**Syntax**

**Definition:**

METHOD Get : HRESULT
VAR_INPUT
   aPoint : Reference To TcVnPoint3_REAL;
END_VAR
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aPoint</td>
<td>Reference To TcVnPoint3_REAL</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Set

Sets the value

**Syntax**

Definition:

```plaintext
METHOD Set : HRESULT
VAR_INPUT
    aPoint : Reference To TcVnPoint3_REAL;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aPoint</td>
<td>Reference To TcVnPoint3_REAL</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

#### 6.3.1.2.13 ITcVnAccess_TcVnRectangle_DINT

Offers an access interface for TcVnRectangle_DINT values.

**Inheritance Hierarchy**

ITcUnknown [399]

ITcVnAccess_TcVnRectangle_DINT
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [246]</td>
<td>ITcVnAccess_TcVnRectangle_DINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [246]</td>
<td>ITcVnAccess_TcVnRectangle_DINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### Get

```plaintext
stRectangle Reference To TcVnRectangle_DINT HRESULT Get
```

Gets the value

### Syntax

**Definition:**

```plaintext
METHOD Get : HRESULT
VAR_INPUT
  stRectangle : Reference To TcVnRectangle_DINT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stRectangle</td>
<td>Reference To TcVnRectangle_DINT [221]</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Set

```plaintext
stRectangle Reference To TcVnRectangle_DINT HRESULT Set
```

Sets the value

### Syntax

**Definition:**

```plaintext
METHOD Set : HRESULT
VAR_INPUT
  stRectangle : Reference To TcVnRectangle_DINT;
END_VAR
```
**METHOD** Set : HRESULT

**VAR_INPUT**

stRectangle : Reference To TcVnRectangle_DINT;

END_VAR

## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stRectangle</td>
<td>Reference To TcVnRectangle_DINT [221]</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

### 6.3.1.2.14  ITCvNAccess_TcVnVector2_DINT

Offers an access interface for TcVnVector2_DINT values.

#### Inheritance Hierarchy

| ITcUnknown [399] | ITCvNAccess_TcVnVector2_DINT |

#### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef</td>
<td>ITCUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface</td>
<td>ITCUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease</td>
<td>ITCUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get</td>
<td>ITCvNAccess_TcVnVector2_DINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set</td>
<td>ITCvNAccess_TcVnVector2_DINT</td>
<td>Sets the value.</td>
</tr>
</tbody>
</table>

#### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### Get

```plaintext
Get

Get

aVector Reference To TcVnVector2_DINT HRESULT Get
```

Gets the value

#### Syntax

Definition:
METHOD Get : HRESULT
VAR_INPUT
   aVector : Reference To TcVnVector2_DINT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector2_DINT [153]</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Set

Sets the value.

Syntax

Definition:
METHOD Set : HRESULT
VAR_INPUT
   aVector : Reference To TcVnVector2_DINT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector2_DINT [153]</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.3.1.2.15 ITcVnAccess_TcVnVector2_INT

Offers an access interface for TcVnVector2_INT values.

Inheritance Hierarchy

ITcUnknown [399]
   ITcVnAccess_TcVnVector2_INT
Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [249]</td>
<td>ITcVnAccess_TcVnVector2_INT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [249]</td>
<td>ITcVnAccess_TcVnVector2_INT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

Get

```plaintext
Get

aVector : Reference To TcVnVector2_INT HRESULT Get
```

Gets the value

Syntax

Definition:

```plaintext
METHOD Get : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector2_INT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector2_INT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Set

```plaintext
Set

aVector : Reference To TcVnVector2_INT HRESULT Set
```

Sets the value

Syntax

Definition:
**API reference**

**METHOD Set : HRESULT**

```plaintext
VAR_INPUT
    aVector : Reference To TcVnVector2_INT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector2_INT [153]</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

#### 6.3.1.2.16  ITCVnAccess_TcVnVector2_REAL

Offers an access interface for TcVnVector2_REAL values.

**Inheritance Hierarchy**

ITcUnknown [399]

  ITCVnAccess_TcVnVector2_REAL

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITCUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITCUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITCUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [250]</td>
<td>ITCVnAccess_TcVnVector2_REAL</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [251]</td>
<td>ITCVnAccess_TcVnVector2_REAL</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### Get

```plaintext
Get
```

Gets the value

### Syntax

**Definition:**

```plaintext
METHOD Get : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector2_REAL;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector2_REAL</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Set**

Sets the value

**Syntax**

Definition:

```c
METHOD Set : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector2_REAL;
END_VAR
```

**Inheritance Hierarchy**

ITcUnknown [399]

ITcVnAccess_TcVnVector2_SINT

6.3.1.2.17  ITcVnAccess_TcVnVector2_SINT

Offers an access interface for TcVnVector2_SINT values.
Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [252]</td>
<td>ITcVnAccess_TcVnVector2_SINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [252]</td>
<td>ITcVnAccess_TcVnVector2_SINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

Get

- **Get**
  
  ```
  aVector : Reference To TcVnVector2_SINT
  Get
  ```

  Gets the value

Syntax

**Definition:**

```plaintext
METHOD Get : HRESULT
VAR_INPUT
  aVector : Reference To TcVnVector2_SINT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector2_SINT [153]</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

Return value

**HRESULT [135]**

Set

- **Set**

  ```
  aVector : Reference To TcVnVector2_SINT
  Set
  ```

  Sets the value.

Syntax

**Definition:**
**METHOD Set : HRESULT**

```plaintext
VAR_INPUT
  aVector : Reference To TcVnVector2_SINT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector2_SINT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

### Return value

HRRESULT [135]

---

### 6.3.1.2.18 ITCvNAccess_TcVnVector2_UINT

Offers an access interface for TcVnVector2_UINT values.

#### Inheritance Hierarchy

- ITCUnknown [399]
  - ITCvNAccess_TcVnVector2_UINT

#### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITCUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITCUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITCUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [253]</td>
<td>ITCvNAccess_TcVnVector2_UINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [254]</td>
<td>ITCvNAccess_TcVnVector2_UINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

#### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

#### Get

```plaintext
aVector Reference To TcVnVector2_UINT HRESULT Get
```

Gets the value

#### Syntax

**Definition:**

```plaintext
METHOD Get : HRESULT
VAR_INPUT
  aVector : Reference To TcVnVector2_UINT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector2_UINT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Set**

Sets the value.

**Syntax**

Definition:

METHOD Set : HRESULT
VAR_INPUT
  aVector : Reference To TcVnVector2_UINT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector2_UINT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

6.3.1.2.19   ITcVnAccess_TcVnVector2_USINT

Offers an access interface for TcVnVector2_USINT values.

**Inheritance Hierarchy**

ITcUnknown [399]
ITcVnAccess_TcVnVector2_USINT
Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [255]</td>
<td>ITcVnAccess_TcVnVector2_USINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [255]</td>
<td>ITcVnAccess_TcVnVector2_USINT</td>
<td>Sets the value.</td>
</tr>
</tbody>
</table>

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

Get

```
METHOD Get : HRESULT
VAR_INPUT
  aVector : Reference To TcVnVector2_USINT;  
END_VAR
```

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
  aVector : Reference To TcVnVector2_USINT;  
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector2_USINT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Set

```
METHOD Set : HRESULT
VAR_INPUT
  aVector : Reference To TcVnVector2_USINT;  
END_VAR
```

Sets the value.

Syntax

Definition:
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector2_USINT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

6.3.1.2.20 **ITcVnAccess_TcVnVector3_INT**

Offers an access interface for TcVnVector3_INT values.

#### Inheritance Hierarchy

ITcUnknown [399]

ITcVnAccess_TcVnVector3_INT

#### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [256]</td>
<td>ITcVnAccess_TcVnVector3_INT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [257]</td>
<td>ITcVnAccess_TcVnVector3_INT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

#### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### Get

Get

*aVector  Reference To TcVnVector3_INT  HRESULT  Get*

Gets the value

#### Syntax

Definition:
METHOD Get : HRESULT
VAR_INPUT
  aVector : Reference To TcVnVector3_INT;
END_VAR

TextStyle Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector3_INT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

TextStyle Return value

Return value
 HRESULT [135]

*Set*

Syntax

Definition:

METHOD Set : HRESULT
VAR_INPUT
  aVector : Reference To TcVnVector3_INT;
END_VAR

TextStyle Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector3_INT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

TextStyle Return value

Return value
 HRESULT [135]

6.3.1.2.21 ITCVnAccess_TcVnVector3_REAL

Offers an access interface for TcVnVector3_REAL values.

Inheritance Hierarchy

ITcUnknown [399]
  ITCVnAccess_TcVnVector3_REAL
Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [258]</td>
<td>ITcVnAccess_TcVnVector 3_REAL</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [258]</td>
<td>ITcVnAccess_TcVnVector 3_REAL</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

Get

```
Get

aVector Reference To TcVnVector3_REAL HRESULT Get
```

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
  aVector : Reference To TcVnVector3_REAL;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector3_REAL [153]</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Set

```
Set

aVector Reference To TcVnVector3_REAL HRESULT Set
```

Sets the value

Syntax

Definition:
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector3_REAL</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

### 6.3.1.2.22 ITCvAccess_TcVnVector3_SINT

Offers an access interface for TcVnVector3_SINT values.

**Inheritance Hierarchy**

ITcUnknown [399]

- ITCvAccess_TcVnVector3_SINT

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [259]</td>
<td>ITCvAccess_TcVnVector3_SINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [260]</td>
<td>ITCvAccess_TcVnVector3_SINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**Get**

```plaintext
Get

aVector Reference To TcVnVector3_SINT HRESULT Get
```

Gets the value

**Syntax**

**Definition:**

METHOD Get : HRESULT

VAR_INPUT

- aVector : Reference To TcVnVector3_SINT;

END_VAR

---

**METHOD Set : HRESULT**

VAR_INPUT

- aVector : Reference To TcVnVector3_REAL;

END_VAR
# Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector3_SINT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

# Return value

HRESULT [135]

## Set

Sets the value

### Syntax

**Definition:**

```plaintext
METHOD Set : HRESULT
VAR_INPUT
  aVector : Reference To TcVnVector3_SINT;
END_VAR
```

# Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector3_SINT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

# Return value

HRESULT [135]

## ITcVnAccess_TcVnVector3_UINT

Offers an access interface for TcVnVector3_UINT values.

### Inheritance Hierarchy

- ITcUnknown [399]
- ITcVnAccess_TcVnVector3_UINT
Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get</td>
<td>ITcVnAccess_TcVnVector 3_UINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set</td>
<td>ITcVnAccess_TcVnVector 3_UINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

Get

```
Get

aVector Reference To TcVnVector3_UINT HRESULT Get
```

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
  aVector : Reference To TcVnVector3_UINT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector3_UINT [153]</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Set

```
Set

aVector Reference To TcVnVector3_UINT HRESULT Set
```

Sets the value

Syntax

Definition:
METHOD Set : HRESULT
VAR_INPUT
  aVector : Reference To TcVnVector3_UINT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector3_UINT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.3.1.2.24  ITCvVnAccess_TcVnVector3_USINT

Offers an access interface for TcVnVector3_USINT values.

Inheritance Hierarchy

ITcUnknown [399]
  ITCvVnAccess_TcVnVector3_USINT

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITCUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITCUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITCUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [262]</td>
<td>ITCvVnAccess_TcVnVector3_USINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [263]</td>
<td>ITCvVnAccess_TcVnVector3_USINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

Get

aVector  Reference To TcVnVector3_USINT \texttt{Get}

Gets the value

Syntax

Definition:
METHOD Get : HRESULT
VAR_INPUT
  aVector : Reference To TcVnVector3_USINT;
END_VAR
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector3_USINT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Set

```
METHOD Set : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector3_USINT;
END_VAR
```

Sets the value

Syntax

Definition:

6.3.1.2.25 ITcVnAccess_TcVnVector4_DINT

Offers an access interface for TcVnVector4_DINT values.

Inheritance Hierarchy

ITcUnknown [399]
ITcVnAccess_TcVnVector4_DINT
### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [264]</td>
<td>ITcVnAccess_TcVnVector4_DINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [264]</td>
<td>ITcVnAccess_TcVnVector4_DINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### Get

- **Get**

  ```
  aVector Reference To TcVnVector4_DINT HRESULT Get
  ```

  Gets the value

### Syntax

**Definition:**

```plaintext
METHOD Get : HRESULT
VAR_INPUT
  aVector : Reference To TcVnVector4_DINT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_DINT [153]</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

### Return value

**HRESULT** [135]

### Set

- **Set**

  ```
  aVector Reference To TcVnVector4_DINT HRESULT Set
  ```

  Sets the value

### Syntax

**Definition:**

```plaintext
METHOD Set : HRESULT
VAR_INPUT
  aVector : Reference To TcVnVector4_DINT;
END_VAR
```
METHOD Set : HRESULT  
VAR_INPUT  
  aVector : Reference To TcVnVector4_DINT;  
END_VAR

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_INT [153]</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### 6.3.1.2.26 ITCvNAccess_TcVnVector4_INT

Offers an access interface for TcVnVector4_INT values.

### Inheritance Hierarchy

ITcUnknown [399]  
  ITCvNAccess_TcVnVector4_INT

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITCUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITCUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITCUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [265]</td>
<td>ITCvNAccess_TcVnVector4_INT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [266]</td>
<td>ITCvNAccess_TcVnVector4_INT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### Get

GET  

**aVector**  
Reference To TcVnVector4_INT  
HRESULT  

Gets the value

### Syntax

**Definition:**

METHOD Get : HRESULT  
VAR_INPUT  
  aVector : Reference To TcVnVector4_INT;  
END_VAR
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_INT <code>[153]</code></td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

### Return value

HRESULT `[135]`

### Set

**Set**

Sets the value

**Syntax**

Definition:

```plaintext
METHOD Set : HRESULT
VAR_INPUT
  aVector : Reference To TcVnVector4_INT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_INT <code>[153]</code></td>
<td>The value to set</td>
</tr>
</tbody>
</table>

### Return value

HRESULT `[135]`

### 6.3.1.2.27  ITcVnAccess_TcVnVector4_LREAL

Offers an access interface for TcVnVector4_LREAL values.

**Inheritance Hierarchy**

ITcUnknown `[399]`

ITcVnAccess_TcVnVector4_LREAL
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [267]</td>
<td>ITcVnAccess_TcVnVector4_LREAL</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [267]</td>
<td>ITcVnAccess_TcVnVector4_LREAL</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## Get

```plaintext
Get

aVector Reference To TcVnVector4_LREAL HRESULT Get
```

Gets the value

### Syntax

**Definition:**

```plaintext
METHOD Get : HRESULT
VAR_INPUT
  aVector : Reference To TcVnVector4_LREAL;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

## Set

```plaintext
Set

aVector Reference To TcVnVector4_LREAL HRESULT Set
```

Sets the value

### Syntax

**Definition:**
METHOD Set : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector4_LREAL;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

### 6.3.1.2.28  ITcVnAccess_TcVnVector4_SINT

Offers an access interface for TcVnVector4_SINT values.

**Inheritance Hierarchy**

ITcUnknown [399]
    ITcVnAccess_TcVnVector4_SINT

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [268]</td>
<td>ITcVnAccess_TcVnVector4_SINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [269]</td>
<td>ITcVnAccess_TcVnVector4_SINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**Get**

```
Get
```

Gets the value

**Syntax**

Definition:
METHOD Get : HRESULT
VAR_INPUT
  aVector : Reference To TcVnVector4_SINT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_SINT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Set**

Sets the value

**Syntax**

Definition:

METHOD Set : HRESULT
VAR_INPUT
  aVector : Reference To TcVnVector4_SINT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_SINT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

6.3.1.2.29  ITcVnAccess_TcVnVector4_UINT

Offers an access interface for TcVnVector4_UINT values.

**Inheritance Hierarchy**

ITcUnknown [399]
  ITcVnAccess_TcVnVector4_UINT
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get</td>
<td>ITcVnAccess_TcVnVector4_UINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set</td>
<td>ITcVnAccess_TcVnVector4_UINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

## System Requirements

### Development environment
- TwinCAT V3.1.4024.17 or later

### Target platform
- PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU

### PLC libraries to include
- Tc3_Vision

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_UINT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Set

Sets the value

### Syntax

**Definition:**

```c
METHOD Get : HRESULT
VAR_INPUT
  aVector : Reference To TcVnVector4_UINT;
END_VAR
```
METHOD Set : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector4_UINT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_UINT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.3.1.2.30  ITCvnAccess_TcVnVector4_USINT

Offers an access interface for TcVnVector4_USINT values.

Inheritance Hierarchy

ITcUnknown [399]
   ITCvnAccess_TcVnVector4_USINT

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITCUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITCUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITCUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [271]</td>
<td>ITCvnAccess_TcVnVector4_USINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [272]</td>
<td>ITCvnAccess_TcVnVector4_USINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

Get

Get

aVector  Reference To TcVnVector4_USINT  HRESULT  Get

 Gets the value

Syntax

Definition:

METHOD Get : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector4_UINT;
END_VAR
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_USINT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Set

Sets the value

Syntax

Definition:

METHOD Set : HRESULT
VAR_INPUT
   aVector : Reference To TcVnVector4_USINT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_USINT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.3.1.2.31 ITcVnAccess_UDINT

Offers an access interface for UDINT values.

Inheritance Hierarchy

ITcUnknown [399]
ITcVnAccess_UDINT
Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [273]</td>
<td>ITcVnAccess_UDINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [273]</td>
<td>ITcVnAccess_UDINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

Get

```
METHOD Get : HRESULT
VAR_INPUT
    nValue : Reference To UDINT;
END_VAR
```

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    nValue : Reference To UDINT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nValue</td>
<td>Reference To UDINT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Set

```
METHOD Set : HRESULT
VAR_INPUT
    nValue : UDINT;
END_VAR
```

Sets the value

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    nValue : UDINT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nValue</td>
<td>UDINT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## 6.3.1.2.32  ITcVnAccess_UINT

Offers an access interface for UINT values.

### Inheritance Hierarchy

- ITcUnknown [399]
- ITcVnAccess_UINT

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [274]</td>
<td>ITcVnAccess_UINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [275]</td>
<td>ITcVnAccess_UINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## Get

### Syntax

**Definition:**

METHOD Get : HRESULT
VAR_INPUT
  nValue : Reference To UINT;
END_VAR

Gets the value

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nValue</td>
<td>Reference To UINT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Set

Sets the value

Syntax

Definition:

METHOD Set : HRESULT
VAR_INPUT
    nValue : UINT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nValue</td>
<td>UINT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.3.1.2.33 ITcVnAccess_ULINT

Offers an access interface for ULINT values.

Inheritance Hierarchy

ITcUnknown [399]
    ITcVnAccess_ULINT

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [276]</td>
<td>ITcVnAccess_ULINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [276]</td>
<td>ITcVnAccess_ULINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
Get

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nValue</td>
<td>Reference To ULINT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

Syntax

Definition:

```c
METHOD Get : HRESULT
VAR_INPUT
  nValue : Reference To ULINT;
END_VAR
```

Inputs

Returns the value

Return value

HRESULT → 135

Set

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nValue</td>
<td>ULINT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

Syntax

Definition:

```c
METHOD Set : HRESULT
VAR_INPUT
  nValue : ULINT;
END_VAR
```

Inputs

The value to set

Return value

HRESULT → 135

6.3.1.2.34 ITcVnAccess_USINT

Offers an access interface for USINT values.
Inheritance Hierarchy

ITcUnknown [ spos 399]  
  ITcVnAccess_USINT

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [ spos 399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [ spos 399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [ spos 400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [ spos 277]</td>
<td>ITcVnAccess_USINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [ spos 277]</td>
<td>ITcVnAccess_USINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

Get

Get

<table>
<thead>
<tr>
<th>nValue</th>
<th>Reference To USINT</th>
<th>HRESULT</th>
<th>Get</th>
</tr>
</thead>
</table>

Gets the value

Syntax

Definition:

```c
METHOD Get : HRESULT
VAR_INPUT
  nValue : Reference To USINT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nValue</td>
<td>Reference To USINT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

Return value

HRESULT [ spos 135]

Set

Set

<table>
<thead>
<tr>
<th>nValue</th>
<th>USINT</th>
<th>HRESULT</th>
<th>Set</th>
</tr>
</thead>
</table>

Sets the value
Syntax

Definition:

METHOD Set : HRESULT
VAR_INPUT
  nValue : USINT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nValue</td>
<td>USINT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.3.1.3 ITCvNRandomAccess

The interface ITCvNRandomAccess is for accessing any elements of a container [144] and is provided by an ITCvNRandomAccessIterator [342]. Unlike ITCvAccess [228], any element in a container can be addressed.

6.3.1.3.1 ITCvNRandomAccess_DINT

Offers a random access interface for DINT values.

Inheritance Hierarchy

ITcUnknown [399]
  ITCvNAccess_DINT [228]
    ITCvNRandomAccess_DINT

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITCUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface</td>
<td>ITCUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITCUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [228]</td>
<td>ITCvNAccess_DINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [229]</td>
<td>ITCvNAccess_DINT</td>
<td>Sets the value</td>
</tr>
<tr>
<td>GetAt [279]</td>
<td>ITCvNRandomAccess_DINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt [279]</td>
<td>ITCvNRandomAccess_DINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
GetAt

### Syntax

**Definition:**

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    nValue : Reference To DINT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>nValue</td>
<td>Reference To DINT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

### Return value

**HRESULT [135]**

SetAt

### Syntax

**Definition:**

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    nValue : DINT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>nValue</td>
<td>DINT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

### Return value

**HRESULT [135]**
6.3.1.3.2  ITCvRandomAccess_INT

Offers a random access interface for INT values

Inheritance Hierarchy

ITcUnknown [399]
   ITCvAccess_INT [229]
      ITCvRandomAccess_INT

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef</td>
<td>ITCUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface</td>
<td>ITCUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease</td>
<td>ITCUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get</td>
<td>ITCvAccess_INT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set</td>
<td>ITCvAccess_INT</td>
<td>Sets the value</td>
</tr>
<tr>
<td>GetAt</td>
<td>ITCvRandomAccess_INT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt</td>
<td>ITCvRandomAccess_INT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

GetAt

\[ \text{GetAt} \]

\[ \text{nOffset} \quad \text{LINT} \quad \text{nValue} \quad \text{Reference To INT} \]

\[ \text{HRESULT} \quad \text{GetAt} \]

Gets the value

Syntax

Definition:

\[
\text{METHOD GetAt : HRESULT}
\]

\[
\text{VAR_INPUT}
\]

\[
\text{nOffset : LINT;}
\]

\[
\text{nValue : Reference To INT;}
\]

\[
\text{END_VAR}
\]

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>nValue</td>
<td>Reference To INT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

SetAt

Sets the value

Syntax

Definition:

METHOD SetAt : HRESULT
  VAR_INPUT
    nOffset : LINT;
    nValue : INT;
  END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>nValue</td>
<td>INT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.3.1.3.3 ITcVnRandomAccess_LREAL

Offers a random access interface for LREAL values.

Inheritance Hierarchy

ITcUnknown [399]
  ITcVnAccess_LREAL [231]
    ITcVnRandomAccess_LREAL

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get</td>
<td>ITcVnAccess_LREAL</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set</td>
<td>ITcVnAccess_LREAL</td>
<td>Sets the value</td>
</tr>
<tr>
<td>GetAt</td>
<td>ITcVnRandomAccess_LREAL</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt</td>
<td>ITcVnRandomAccess_LREAL</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

GetAt

**Syntax**

**Definition:**

```plaintext
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    fValue : Reference To LREAL;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position.</td>
</tr>
<tr>
<td>fValue</td>
<td>Reference To LREAL</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

SetAt

**Syntax**

**Definition:**

```plaintext
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    fValue : LREAL;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position.</td>
</tr>
<tr>
<td>fValue</td>
<td>LREAL</td>
<td>The value to set</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

6.3.1.3.4 ITCvnRandomAccess_REAL

Offers a random access interface for REAL values.

Inheritance Hierarchy

ITcUnknown [399]
  ITCvAccess_REAL [232]
  ITCvRandomAccess_REAL

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [233]</td>
<td>ITCvAccess_REAL</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [233]</td>
<td>ITCvAccess_REAL</td>
<td>Sets the value</td>
</tr>
<tr>
<td>GetAt [283]</td>
<td>ITCvRandomAccess_REAL</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt [284]</td>
<td>ITCvRandomAccess_REAL</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

GetAt

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    fValue : Reference To REAL;
END_VAR
```

Gets the value

Syntax

Definition:
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>fValue</td>
<td>Reference To REAL</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

SetAt

Sets the value

Syntax

Definition:

METHOD SetAt : HRESULT
VAR_INPUT
  nOffset : LINT;
  fValue : REAL;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>fValue</td>
<td>REAL</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.3.1.3.5 ITcVnRandomAccess_SINT

Offers a random access interface for SINT values

Inheritance Hierarchy

ITcUnknown [399]
  ITcVnAccess_SINT [233]
  ITcVnRandomAccess_SINT
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [234]</td>
<td>ITcVnAccess_SINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [234]</td>
<td>ITcVnAccess_SINT</td>
<td>Sets the value</td>
</tr>
<tr>
<td>GetAt [285]</td>
<td>ITcVnRandomAccess_SI</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt [285]</td>
<td>ITcVnRandomAccess_SI</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## GetAt

### Syntax

**Definition:**

```
METHOD GetAt   : HRESULT
VAR_INPUT
    nOffset : LINT;
    nValue  : Reference To SINT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>nValue</td>
<td>Reference To SINT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

### Return value

**HRESULT [135]**

## SetAt

### Syntax

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    nValue  : SINT;
END_VAR
```

Sets the value
## Syntax

**Definition:**

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    nValue  : SINT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>nValue</td>
<td>SINT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

## 6.3.1.3.6 ITCvRandomAccess_TcVnDMatch

Offers a random access interface for TcVnDMatch values.

### Inheritance Hierarchy

ITcUnknown [399]

ITcVnAccess_TcVnDMatch [235]

ITcVnRandomAccess_TcVnDMatch

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITCUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITCUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITCUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [235]</td>
<td>ITCvVnAccess_TcVnDMatch</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [236]</td>
<td>ITCvVnAccess_TcVnDMatch</td>
<td>Sets the value</td>
</tr>
<tr>
<td>GetAt [287]</td>
<td>ITCvVnRandomAccess_TcVnDMatch</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt [287]</td>
<td>ITCvVnRandomAccess_TcVnDMatch</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
GetAt

Gets the value

Syntax

Definition:

METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    stDMatch : Reference To TcVnDMatch;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position.</td>
</tr>
<tr>
<td>stDMatch</td>
<td>Reference To TcVnDMatch</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

SetAt

Sets the value

Syntax

Definition:

METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    stDMatch : Reference To TcVnDMatch;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position.</td>
</tr>
<tr>
<td>stDMatch</td>
<td>Reference To TcVnDMatch</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]
6.3.1.3.7  ITcVnRandomAccess_TcVnKeyPoint

Offers a random access interface for TcVnKeyPoint values.

Inheritance Hierarchy

ITcUnknown [ 399]
  ITcVnAccess_TcVnKeyPoint [ 236]
  ITcVnRandomAccess_TcVnKeyPoint

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [ 399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [ 399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [ 400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [ 237]</td>
<td>ITcVnAccess_TcVnKeyPoint</td>
<td>Gets the value.</td>
</tr>
<tr>
<td>Set [ 237]</td>
<td>ITcVnAccess_TcVnKeyPoint</td>
<td>Sets the value.</td>
</tr>
<tr>
<td>GetAt [ 288]</td>
<td>ITcVnRandomAccess_TcVnKeyPoint</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt [ 289]</td>
<td>ITcVnRandomAccess_TcVnKeyPoint</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

GetAt

```plaintext
METHOD GetAt : HRESULT
VAR_INPUT
  nOffset    : LINT;
  stKeyPoint : Reference To TcVnKeyPoint;
END_VAR
```

Gets the value

Syntax

Definition:

```plaintext
METHOD GetAt : HRESULT
VAR_INPUT
  nOffset    : LINT;
  stKeyPoint : Reference To TcVnKeyPoint;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>stKeyPoint</td>
<td>Reference To</td>
<td>Returns the value</td>
</tr>
<tr>
<td></td>
<td>TcVnKeyPoint</td>
<td></td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

SetAt

Sets the value

Syntax

Definition:

METHOD SetAt : HRESULT
VAR_INPUT
  nOffset : LINT;
  stKeyPoint : Reference To TcVnKeyPoint;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>stKeyPoint</td>
<td>Reference To TcVnKeyPoint [208]</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.3.1.3.8 ITcVnRandomAccess_TcVnPoint2_DINT

Offers a random access interface for TcVnPoint2_DINT values.

Inheritance Hierarchy

ITcUnknown [399]
  ITcVnAccess_TcVnPoint2_DINT [238]
  ITcVnRandomAccess_TcVnPoint2_DINT
### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [238]</td>
<td>ITcVnAccess_TcVnPoint2_DINT</td>
<td>Gets the value.</td>
</tr>
<tr>
<td>Set [239]</td>
<td>ITcVnAccess_TcVnPoint2_DINT</td>
<td>Sets the value.</td>
</tr>
<tr>
<td>GetAt [290]</td>
<td>ITcVnRandomAccess_TcVnPoint2_DINT</td>
<td>Gets the value.</td>
</tr>
<tr>
<td>SetAt [291]</td>
<td>ITcVnRandomAccess_TcVnPoint2_DINT</td>
<td>Sets the value.</td>
</tr>
</tbody>
</table>

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### GetAt

```plaintext
GetAt

nOffset  LINT   HRESULT  GetAt
aPoint    Reference To TcVnPoint2_DINT
```

Gets the value

### Syntax

**Definition:**

```plaintext
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aPoint : Reference To TcVnPoint2_DINT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aPoint</td>
<td>Reference To TcVnPoint2_DINT [151]</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

### Return value

**HRESULT [135]**
SetAt

Sets the value

Syntax

Definition:

METHOD SetAt : HRESULT
VAR_INPUT
  nOffset : LINT;
  aPoint : Reference To TcVnPoint2_DINT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aPoint</td>
<td>Reference To TcVnPoint2_DINT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.3.1.3.9 ITcVnRandomAccess_TcVnPoint2_LREAL

Offers a random access interface for TcVnPoint2_LREAL values.

Inheritance Hierarchy

ITcUnknown [399]
  ITcVnAccess_TcVnPoint2_LREAL [239]
  ITcVnRandomAccess_TcVnPoint2_LREAL

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [240]</td>
<td>ITcVnAccess_TcVnPoint2_LREAL</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [240]</td>
<td>ITcVnAccess_TcVnPoint2_LREAL</td>
<td>Sets the value</td>
</tr>
<tr>
<td>GetAt [292]</td>
<td>ITcVnRandomAccess_TcVnPoint2_LREAL</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt [292]</td>
<td>ITcVnRandomAccess_TcVnPoint2_LREAL</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

GetAt

```plaintext
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aPoint  : Reference To TcVnPoint2_LREAL;
END_VAR
```

Gets the value

Syntax

Definition:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aPoint</td>
<td>Reference To TcVnPoint2_LREAL</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

SetAt

```plaintext
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aPoint  : Reference To TcVnPoint2_LREAL;
END_VAR
```

Sets the value

Syntax

Definition:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aPoint</td>
<td>Reference To TcVnPoint2_LREAL</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aPoint</td>
<td>Reference To TcVnPoint2_REAL [151]</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

### 6.3.1.3.10 ITcVnRandomAccess_TcVnPoint2_REAL

Offers a random access interface for TcVnPoint2_REAL values.

#### Inheritance Hierarchy

- ITcUnknown [399]
  - ITcVnAccess_TcVnPoint2_REAL [241]
  - ITcVnRandomAccess_TcVnPoint2_REAL

#### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [241]</td>
<td>ITcVnAccess_TcVnPoint2_REAL</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [242]</td>
<td>ITcVnAccess_TcVnPoint2_REAL</td>
<td>Sets the value</td>
</tr>
<tr>
<td>GetAt [293]</td>
<td>ITcVnRandomAccess_TcVnPoint2_REAL</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt [294]</td>
<td>ITcVnRandomAccess_TcVnPoint2_REAL</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

#### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### GetAt

Input:

- **nOffset**: LINT
- **aPoint**: Reference To TcVnPoint2_REAL

Output:

- **HRESULT**
- **GetAt**

Gets the value

#### Syntax

Definition:
### GetAt

**Signature**

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aPoint : Reference To TcVnPoint2_REAL;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aPoint</td>
<td>Reference To TcVnPoint2_REAL</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

### SetAt

Sets the value

**Syntax**

**Definition**

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aPoint : Reference To TcVnPoint2_REAL;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aPoint</td>
<td>Reference To TcVnPoint2_REAL</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

### ITcVnRandomAccess_TcVnPoint3_LREAL

Offers a random access interface for TcVnPoint3_LREAL values.

**Inheritance Hierarchy**

ITcUnknown [399]

ITcVnAccess_TcVnPoint3_LREAL [242]

ITcVnRandomAccess_TcVnPoint3_LREAL [151]
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITCUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITCUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITCUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [243]</td>
<td>ITCnAccess_TcVnPoint3_LREAL</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [243]</td>
<td>ITCnAccess_TcVnPoint3_LREAL</td>
<td>Sets the value</td>
</tr>
<tr>
<td>GetAt [295]</td>
<td>ITCnRandomAccess_TcVnPoint3_LREAL</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt [296]</td>
<td>ITCnRandomAccess_TcVnPoint3_LREAL</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

### GetAt

**Syntax**

Definition:

```plaintext
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aPoint  : Reference To TcVnPoint3_LREAL;
END_VAR
```

** Inputs **

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aPoint</td>
<td>Reference To TcVnPoint3_LREAL [151]</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

** Return value **

HRESULT [135]
SetAt

Sets the value

**Syntax**

**Definition:**

METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aPoint : Reference To TcVnPoint3_LREAL;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aPoint</td>
<td>Reference To TcVnPoint3_LREAL</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

6.3.1.3.12 ITcVnRandomAccess_TcVnPoint3_REAL

Offers a random access interface for TcVnPoint3_REAL values.

**Inheritance Hierarchy**

ITcUnknown [399]

   ITcVnAccess_TcVnPoint3_REAL [244]
   ITcVnRandomAccess_TcVnPoint3_REAL

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get</td>
<td>ITcVnAccess_TcVnPoint3_REAL</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set</td>
<td>ITcVnAccess_TcVnPoint3_REAL</td>
<td>Sets the value</td>
</tr>
<tr>
<td>GetAt</td>
<td>ITcVnRandomAccess_TcVnPoint3_REAL</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt</td>
<td>ITcVnRandomAccess_TcVnPoint3_REAL</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

GetAt

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aPoint : Reference To TcVnPoint3_REAL;
END_VAR
```

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aPoint : Reference To TcVnPoint3_REAL;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aPoint</td>
<td>Reference To TcVnPoint3_REAL</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

SetAt

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aPoint : Reference To TcVnPoint3_REAL;
END_VAR
```

Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aPoint : Reference To TcVnPoint3_REAL;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aPoint</td>
<td>Reference To TcVnPoint3_REAL [151]</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

### 6.3.1.3.13 ITcVnRandomAccess_TcVnRectangle_DINT

Offers a random access interface for TcVnRectangle_DINT values.

**Inheritance Hierarchy**

ITcUnknown [399]

ITcVnAccess_TcVnRectangle_DINT [245]

ITcVnRandomAccess_TcVnRectangle_DINT

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [246]</td>
<td>ITcVnAccess_TcVnRectangl e_DINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [246]</td>
<td>ITcVnAccess_TcVnRectangl e_DINT</td>
<td>Sets the value</td>
</tr>
<tr>
<td>GetAt [298]</td>
<td>ITcVnRandomAccess_TcVnRec tangle_DINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt [299]</td>
<td>ITcVnRandomAccess_TcVnRec tangle_DINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**GetAt**

```
GetAt
nOffset LINT HRESULT GetAt
stRectangle Reference To TcVnRectangle_DINT
```

Gets the value

**Syntax**

Definition:
**METHOD GetAt : HRESULT**

```c
VAR_INPUT
    nOffset     : LINT;
    stRectangle : Reference To TcVnRectangle_DINT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>stRectangle</td>
<td>Reference To TcVnRectangle_DINT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

**SetAt**

Sets the value

**Syntax**

Definition:

```c
METHOD SetAt : HRESULT
    VAR_INPUT
        nOffset     : LINT;
        stRectangle : Reference To TcVnRectangle_DINT;
    END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>stRectangle</td>
<td>Reference To TcVnRectangle_DINT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### 6.3.1.3.14 ITcVnRandomAccess_TcVnVector2_DINT

Offers a random access interface for TcVnVector2_DINT values.
Inheritance Hierarchy

ITcUnknown [399]
  ITcVnAccess_TcVnVector2_DINT [247]
  ITcVnRandomAccess_TcVnVector2_DINT

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [247]</td>
<td>ITcVnAccess_TcVnVector2_DINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [248]</td>
<td>ITcVnAccess_TcVnVector2_DINT</td>
<td>Sets the value</td>
</tr>
<tr>
<td>GetAt [300]</td>
<td>ITcVnRandomAccess_TcVnVector2_DINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt [301]</td>
<td>ITcVnRandomAccess_TcVnVector2_DINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

GetAt

```
METHOD GetAt : HRESULT
VAR_INPUT
  nOffset : LINT;
  aVector : Reference To TcVnVector2_DINT;
END_VAR
```

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
  nOffset : LINT;
  aVector : Reference To TcVnVector2_DINT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector2_DINT [153]</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]
SetAt

Sets the value

Syntax

Definition:

METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector2_DINT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector2_DINT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.3.1.3.15  ITCvnRandomAccess_TcVnVector2_INT

Offers a random access interface for TcVnVector2_INT values.

Inheritance Hierarchy

ITcUnknown [399]
    ITCvnAccess_TcVnVector2_INT [248]
        ITCvnRandomAccess_TcVnVector2_INT

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef</td>
<td>ITCUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface</td>
<td>ITCUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease</td>
<td>ITCUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get</td>
<td>ITCvnAccess_TcVnVector2_INT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set</td>
<td>ITCvnAccess_TcVnVector2_INT</td>
<td>Sets the value.</td>
</tr>
<tr>
<td>GetAt</td>
<td>ITCvnRandomAccess_TcVnVector2_INT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt</td>
<td>ITCvnRandomAccess_TcVnVector2_INT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

GetAt

```plaintext
METHOD GetAt : HRESULT
VAR_INPUT
  nOffset : LINT;
  aVector : Reference To TcVnVector2_INT;
END_VAR

HRESULT GetAt
```

Gets the value

Syntax

Definition:

```plaintext
METHOD GetAt : HRESULT
VAR_INPUT
  nOffset : LINT;
  aVector : Reference To TcVnVector2_INT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector2_INT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

SetAt

```plaintext
METHOD SetAt : HRESULT
VAR_INPUT
  nOffset : LINT;
  aVector : Reference To TcVnVector2_INT;
END_VAR
```

Sets the value

Syntax

Definition:

```plaintext
METHOD SetAt : HRESULT
VAR_INPUT
  nOffset : LINT;
  aVector : Reference To TcVnVector2_INT;
END_VAR
```
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To</td>
<td>The value to set</td>
</tr>
<tr>
<td></td>
<td>TcVnVector2_</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INT [153]</td>
<td></td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.3.1.3.16  ITcVnRandomAccess_TcVnVector2_REAL

Offers a random access interface for TcVnVector2_REAL values.

Inheritance Hierarchy

ITcUnknown [399]

ITcVnAccess_TcVnVector2_REAL [250]

ITcVnRandomAccess_TcVnVector2_REAL

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get</td>
<td>ITcVnAccess_TcVnVector2_REAL</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set</td>
<td>ITcVnAccess_TcVnVector2_REAL</td>
<td>Sets the value</td>
</tr>
<tr>
<td>GetAt</td>
<td>ITcVnRandomAccess_TcVnVector2_REAL</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt</td>
<td>ITcVnRandomAccess_TcVnVector2_REAL</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

GetAt

```
GetAt
nOffset  LINT  HRESULT  GetAt
aVector  Reference To TcVnVector2_REAL
```

Gets the value

Syntax

Definition:
**GetAt**

**METHOD GetAt : HRESULT**

**VAR_INPUT**

- `nOffset : LINT;`
- `aVector : Reference To TcVnVector2_REAL;`

END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To</td>
<td>Returns the value</td>
</tr>
<tr>
<td></td>
<td>TcVnVector2_REAL</td>
<td></td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**SetAt**

**METHOD SetAt : HRESULT**

**VAR_INPUT**

- `nOffset : LINT;`
- `aVector : Reference To TcVnVector2_REAL;`

END_VAR

**Syntax**

Definition:

**METHOD SetAt : HRESULT**

**VAR_INPUT**

- `nOffset : LINT;`
- `aVector : Reference To TcVnVector2_REAL;`

END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To</td>
<td>The value to set</td>
</tr>
<tr>
<td></td>
<td>TcVnVector2_REAL</td>
<td></td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

6.3.1.3.17  **ITcVnRandomAccess_TcVnVector2_SINT**

Offers a random access interface for TcVnVector2_SINT values.

**Inheritance Hierarchy**

**ITcUnknown [399]**

- **ITcVnAccess_TcVnVector2_SINT [251]**
  - **ITcVnRandomAccess_TcVnVector2_SINT**
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITCUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITCUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITCUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [252]</td>
<td>ITCvAccessTcVnVector2_SINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [252]</td>
<td>ITCvAccessTcVnVector2_SINT</td>
<td>Sets the value</td>
</tr>
<tr>
<td>GetAt [305]</td>
<td>ITCvRandomAccessTcVnVector2_SINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt [306]</td>
<td>ITCvRandomAccessTcVnVector2_SINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### GetAt

**Syntax**

**Definition:**

```plaintext
METHOD GetAt : HRESULT
VAR_INPUT
   nOffset : LINT;
   aVector : Reference To TcVnVector2_SINT;
END_VAR
```

**GetAt**

- `nOffset` **LINT**: Offset to the current position
- `aVector` **Reference To TcVnVector2_SINT**: Returns the value

**Return value**

- **HRESULT [135]**

---

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector2_SINT [153]</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>
**SetAt**

Sets the value

**Syntax**

Definition:

```plaintext
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector2_SINT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector2_SINT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

---

**6.3.1.3.18 ITcVnRandomAccess_TcVnVector2_UINT**

Offers a random access interface for TcVnVector2_UINT values.

**Inheritance Hierarchy**

ITcUnknown [399]

ITcVnAccess_TcVnVector2_UINT [253]

ITcVnRandomAccess_TcVnVector2_UINT

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [253]</td>
<td>ITcVnAccess_TcVnVector2_UINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [254]</td>
<td>ITcVnAccess_TcVnVector2_UINT</td>
<td>Sets the value</td>
</tr>
<tr>
<td>GetAt [307]</td>
<td>ITcVnRandomAccess_TcVnVector2_UINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt [307]</td>
<td>ITcVnRandomAccess_TcVnVector2_UINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

GetAt

Syntax

Definition:

\[
\text{METHOD GetAt : HRESULT} \\
\text{VAR_INPUT} \\
\quad \text{nOffset : LINT;} \\
\quad \text{aVector : Reference To TcVnVector2_UINT;} \\
\text{END_VAR}
\]

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector2_UINT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

SetAt

Syntax

Definition:

\[
\text{METHOD SetAt : HRESULT} \\
\text{VAR_INPUT} \\
\quad \text{nOffset : LINT;} \\
\quad \text{aVector : Reference To TcVnVector2_UINT;} \\
\text{END_VAR}
\]
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To</td>
<td>The value to set</td>
</tr>
<tr>
<td></td>
<td>TcVnVector2_UINT</td>
<td></td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

### 6.3.1.3.19 ITcVnRandomAccess_TcVnVector2_USINT

Offers a random access interface for TcVnVector2_USINT values.

#### Inheritance Hierarchy

ITcUnknown [399]

- ITcVnAccess_TcVnVector2_USINT [254]
- ITcVnRandomAccess_TcVnVector2_USINT

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [255]</td>
<td>ITcVnAccess_TcVnVector2_USINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [255]</td>
<td>ITcVnAccess_TcVnVector2_USINT</td>
<td>Sets the value</td>
</tr>
<tr>
<td>GetAt [308]</td>
<td>ITcVnRandomAccess_TcVnVector2_USINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt [309]</td>
<td>ITcVnRandomAccess_TcVnVector2_USINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### GetAt

```
GetAt
nOffset LINT                   HRESULT GetAt
aVector Reference To TcVnVector2_USINT
```

Gets the value

### Syntax

Definition:
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector2_USINT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

---

**SetAt**

Sets the value

**Syntax**

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector2_USINT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector2_USINT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

---

**6.3.1.3.20 ITcVnRandomAccess_TcVnVector3_INT**

Offers a random access interface for TcVnVector3_INT values.
Inheritance Hierarchy

ITcUnknown [399]
  ITcVnAccess_TcVnVector3_INT [256]
  ITcVnRandomAccess_TcVnVector3_INT

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [256]</td>
<td>ITcVnAccess_TcVnVector3_INT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [257]</td>
<td>ITcVnAccess_TcVnVector3_INT</td>
<td>Sets the value</td>
</tr>
<tr>
<td>GetAt [310]</td>
<td>ITcVnRandomAccess_TcVnVector3_INT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt [311]</td>
<td>ITcVnRandomAccess_TcVnVector3_INT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

GetAt

![GetAt](image)

Gets the value

Syntax

Definition:

```plaintext
METHOD GetAt : HRESULT
VAR_INPUT
  nOffset : LINT;
  aVector : Reference To TcVnVector3_INT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector3_INT [153]</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]
SetAt

Sets the value

Syntax

Definition:

METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector3_INT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector3_INT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.3.1.3.21 ITcVnRandomAccess_TcVnVector3_REAL

Offers a random access interface for TcVnVector3_REAL values.

Inheritance Hierarchy

ITcUnknown [399]
    ITcVnAccess_TcVnVector3_REAL [257]
    ITcVnRandomAccess_TcVnVector3_REAL

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get</td>
<td>ITcVnAccess_TcVnVector3_REAL</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set</td>
<td>ITcVnAccess_TcVnVector3_REAL</td>
<td>Sets the value</td>
</tr>
<tr>
<td>GetAt</td>
<td>ITcVnRandomAccess_TcVnVector3_REAL</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt</td>
<td>ITcVnRandomAccess_TcVnVector3_REAL</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

GetAt

Gets the value

Syntax

Definition:

METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector3_REAL;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector3_REAL</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

Return value

HRESULT

SetAt

Sets the value

Syntax

Definition:

METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector3_REAL;
END_VAR
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector3_REAL</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.3.1.3.22 ITcVnRandomAccess_TcVnVector3_SINT

Offers a random access interface for TcVnVector3_SINT values.

Inheritance Hierarchy

[ITcUnknown [399]]

[ITcVnAccess_TcVnVector3_SINT [259]]

[ITcVnRandomAccess_TcVnVector3_SINT]

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get</td>
<td>ITcVnAccess_TcVnVector3_SINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set</td>
<td>ITcVnAccess_TcVnVector3_SINT</td>
<td>Sets the value</td>
</tr>
<tr>
<td>GetAt</td>
<td>ITcVnRandomAccess_TcVnVector3_SINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt</td>
<td>ITcVnRandomAccess_TcVnVector3_SINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

GetAt

```plaintext
GetAt
    nOffset LINT
    aVector Reference To TcVnVector3_SINT
```

Gets the value

Syntax

Definition:
**METHOD GetAt** : HRESULT  
VAR_INPUT  
   nOffset : LINT;  
   aVector : Reference To TcVnVector3_SINT;  
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector3_SINT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

SetAt

Sets the value

**Syntax**

**Definition:**

METHOD SetAt : HRESULT  
VAR_INPUT  
   nOffset : LINT;  
   aVector : Reference To TcVnVector3_SINT;  
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector3_SINT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

6.3.1.3.23 ITcVnRandomAccess_TcVnVector3_UINT

Offers a random access interface for TcVnVector3_UINT values.

**Inheritance Hierarchy**

ITcUnknown [399]  
   ITcVnAccess_TcVnVector3_UINT [260]  
   ITcVnRandomAccess_TcVnVector3_UINT
### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [261]</td>
<td>ITcVnAccess_TcVnVector3_UINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [261]</td>
<td>ITcVnAccess_TcVnVector3_UINT</td>
<td>Sets the value</td>
</tr>
<tr>
<td>GetAt [315]</td>
<td>ITcVnRandomAccess_TcVnVector3_UINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt [316]</td>
<td>ITcVnRandomAccess_TcVnVector3_UINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

#### GetAt

```plaintext
 GetAt

 nOffset  LINT  
 aVector  Reference To TcVnVector3_UINT

HRESULT GetAt
```

Gets the value

### Syntax

**Definition:**

```plaintext
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector3_UINT;
END_VAR
```

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector3_UINT [153]</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

#### Return value

HRESULT [135]
SetAt

Sets the value

Syntax

Definition:

METHOD SetAt : HRESULT
VAR_INPUT
  nOffset : LINT;
  aVector : Reference To TcVnVector3_UINT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector3_UINT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

Return value

HRESULT

Inheritance Hierarchy

ITcVnRandomAccess_TcVnVector3_USINT

Offers a random access interface for TcVnVector3_USINT values.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef</td>
<td>ITCUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface</td>
<td>ITCUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease</td>
<td>ITCUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get</td>
<td>ITCVnAccess_TcVnVector3_USINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set</td>
<td>ITCVnAccess_TcVnVector3_USINT</td>
<td>Sets the value</td>
</tr>
<tr>
<td>GetAt</td>
<td>ITCVnRandomAccess_TcVnVector3_USINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt</td>
<td>ITCVnRandomAccess_TcVnVector3_USINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

GetAt

```
METHOD GetAt : HRESULT
VAR_INPUT
  nOffset : LINT;
  aVector : Reference To TcVnVector3_USINT;
END_VAR
```

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
  nOffset : LINT;
  aVector : Reference To TcVnVector3_USINT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector3_USINT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

SetAt

```
METHOD SetAt : HRESULT
VAR_INPUT
  nOffset : LINT;
  aVector : Reference To TcVnVector3_USINT;
END_VAR
```

Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
  nOffset : LINT;
  aVector : Reference To TcVnVector3_USINT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_USINT {^[153]}</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT {^[135]}

6.3.1.3.25  ITCvnRandomAccess_TcVnVector4_DINT

Offers a random access interface for TcVnVector4_DINT values.

**Inheritance Hierarchy**

ITcUnknown {^[399]}
  ITCvnAccess_TcVnVector4_DINT {^[263]}
  ITCvnRandomAccess_TcVnVector4_DINT

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef</td>
<td>ITCUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface</td>
<td>ITCUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease</td>
<td>ITCUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get {^[264]}</td>
<td>ITCvnAccess_TcVnVector4_DINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set {^[264]}</td>
<td>ITCvnAccess_TcVnVector4_DINT</td>
<td>Sets the value</td>
</tr>
<tr>
<td>GetAt {^[318]}</td>
<td>ITCvnRandomAccess_TcVnVector4_DINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt {^[319]}</td>
<td>ITCvnRandomAccess_TcVnVector4_DINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**GetAt**

```
GetAt
   nOffset  LINT
   aVector  Reference To TcVnVector4_DINT
   HRSELECT GetAt
```

Gets the value
Syntax

**Definition:**

```vbnet
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector4_DINT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_DINT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**SetAt**

Sets the value

**Syntax**

**Definition:**

```vbnet
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector4_DINT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_DINT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**6.3.1.3.26 ITCvRandomAccess_TcVnVector4_INT**

Offers a random access interface for TcVnVector4_INT values.
Inheritance Hierarchy

ITcUnknown [➔ 399]
  ITcVnAccess_TcVnVector4_INT [➔ 265]
  ITcVnRandomAccess_TcVnVector4_INT

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [➔ 399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [➔ 399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [➔ 400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [➔ 265]</td>
<td>ITcVnAccess_TcVnVector4_INT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [➔ 266]</td>
<td>ITcVnAccess_TcVnVector4_INT</td>
<td>Sets the value</td>
</tr>
<tr>
<td>GetAt [➔ 320]</td>
<td>ITcVnRandomAccess_TcVnVector4_INT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt [➔ 321]</td>
<td>ITcVnRandomAccess_TcVnVector4_INT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

GetAt

GetAt

\[
\text{GetAt}(\text{nOffset: LINT, aVector: Reference To TcVnVector4_INT}) \rightarrow \text{HRESULT GetAt}
\]

Gets the value

Syntax

Definition:

```c
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector4_INT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_INT [➔ 153]</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

Return value

HRESULT [➔ 135]
SetAt

Sets the value

Syntax

Definition:

METHOD SetAt : HRESULT
VAR_INPUT
   nOffset : LINT;
   aVector : Reference To TcVnVector4_INT;
END_VAR

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_INT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.3.1.3.27 ITcVnRandomAccess_TcVnVector4_LREAL

Offers a random access interface for TcVnVector4_LREAL values.

Inheritance Hierarchy

ITcUnknown [399]
   ITcVnAccess_TcVnVector4_LREAL [266]
   ITcVnRandomAccess_TcVnVector4_LREAL

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [267]</td>
<td>ITcVnAccess_TcVnVector4_LREAL</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [267]</td>
<td>ITcVnAccess_TcVnVector4_LREAL</td>
<td>Sets the value</td>
</tr>
<tr>
<td>GetAt [322]</td>
<td>ITcVnRandomAccess_TcVnVector4_LREAL</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt [322]</td>
<td>ITcVnRandomAccess_TcVnVector4_LREAL</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>
**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**GetAt**

```plaintext
nOffset  LINT
aVector   Reference To TcVnVector4_LREAL
GetAt
```

Gets the value

**Syntax**

**Definition:**

METHOD GetAt : HRESULT
VAR_INPUT
  nOffset : LINT;
  aVector : Reference To TcVnVector4_LREAL;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**SetAt**

```plaintext
nOffset  LINT
aVector   Reference To TcVnVector4_LREAL
SetAt
```

Sets the value

**Syntax**

**Definition:**

METHOD SetAt : HRESULT
VAR_INPUT
  nOffset : LINT;
  aVector : Reference To TcVnVector4_LREAL;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

### 6.3.1.3.28 ITcVnRandomAccess_TcVnVector4_SINT

Offers a random access interface for TcVnVector4_SINT values.

#### Inheritance Hierarchy

ITcUnknown [399]

ITcVnAccess_TcVnVector4_SINT [268]

ITcVnRandomAccess_TcVnVector4_SINT

#### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get</td>
<td>ITcVnAccess_TcVnVector4_SINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set</td>
<td>ITcVnAccess_TcVnVector4_SINT</td>
<td>Sets the value</td>
</tr>
<tr>
<td>GetAt</td>
<td>ITcVnRandomAccess_TcVnVector4_SINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt</td>
<td>ITcVnRandomAccess_TcVnVector4_SINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

#### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### GetAt

```
GetAt

---

nOffset    LINT
aVector    Reference To TcVnVector4_SINT

---

HRESULT GetAt
```

Gets the value
API reference

Syntax

**Definition:**

METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector4_SINT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_SINT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**SetAt**

Sets the value

**Syntax**

**Definition:**

METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector4_SINT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_SINT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**6.3.1.3.29 ITcVnRandomAccess_TcVnVector4_UINT**

Offers a random access interface for TcVnVector4_UINT values.
Inheritance Hierarchy

ITcUnknown [399]
  ITcVnAccess_TcVnVector4_UINT [269]
  ITcVnRandomAccess_TcVnVector4_UINT

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [270]</td>
<td>ITcVnAccess_TcVnVector4_UINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [270]</td>
<td>ITcVnAccess_TcVnVector4_UINT</td>
<td>Sets the value</td>
</tr>
<tr>
<td>GetAt [325]</td>
<td>ITcVnRandomAccess_TcVnVector4_UINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt [326]</td>
<td>ITcVnRandomAccess_TcVnVector4_UINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

GetAt

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector4_UINT;
END_VAR
```

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector4_UINT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_UINT [153]</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

Return value

```
HRESULT [135]
```
SetAt

Sets the value

Syntax

Definition:

```c
METHOD SetAt : HRESULT
VAR_INPUT
   nOffset : LINT;
   aVector : Reference To TcVnVector4_UINT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_UINT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.3.1.3.30 ITCvNRandomAccess_TcVnVector4_USINT

Offers a random access interface for TcVnVector4_USINT values.

Inheritance Hierarchy

- ITCUnknown [399]
  - ITCvNAccess_TcVnVector4_USINT [271]
    - ITCvNRandomAccess_TcVnVector4_USINT

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef</td>
<td>ITCUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface</td>
<td>ITCUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease</td>
<td>ITCUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get</td>
<td>ITCvNAccess_TcVnVector4_USINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set</td>
<td>ITCvNAccess_TcVnVector4_USINT</td>
<td>Sets the value</td>
</tr>
<tr>
<td>GetAt</td>
<td>ITCvNRandomAccess_TcVnVector4_USINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt</td>
<td>ITCvNRandomAccess_TcVnVector4_USINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

GetAt

**Definition:**

```c
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector4_USINT;
END_VAR
```

Gets the value

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_USINT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

**Return value**

```
HRESULT [135]
```

SetAt

**Definition:**

```c
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector4_USINT;
END_VAR
```

Sets the value

**Syntax**

**Definition:**

```c
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector4_USINT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_USINT [153]</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

### 6.3.1.3.31 ITCvRandomAccess_UDINT

Offers a random access interface for UDINT values.

#### Inheritance Hierarchy

ITcUnknown [399]

ITcVnAccess_UDINT [272]

ITcVnRandomAccess_UDINT

#### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [273]</td>
<td>ITcVnAccess_UDINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [273]</td>
<td>ITcVnAccess_UDINT</td>
<td>Sets the value</td>
</tr>
<tr>
<td>GetAt [328]</td>
<td>ITcVnRandomAccess_UDINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt [329]</td>
<td>ITcVnRandomAccess_UDINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

#### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### GetAt

```plaintext
GetAt
nOffset LINT
nValue Reference To UDINT
HRESULT GetAt
```

Gets the value

#### Syntax

**Definition:**

328 Version: 1.3 TF7000 - TF7300
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    nValue : Reference To UDINT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>nValue</td>
<td>Reference To UDINT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

SetAt

Sets the value

Syntax

Definition:

METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    nValue : UDINT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>nValue</td>
<td>UDINT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.3.1.3.32 ITCvRandomAccess_UINT

Offers a random access interface for UINT values

Inheritance Hierarchy

ITcUnknown [399]
    ITCvAccess_UINT [274]
    ITCvRandomAccess_UINT
### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [274]</td>
<td>ITcVnAccess_UINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [275]</td>
<td>ITcVnAccess_UINT</td>
<td>Sets the value</td>
</tr>
<tr>
<td>GetAt [330]</td>
<td>ITcVnRandomAccess_UINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt [330]</td>
<td>ITcVnRandomAccess_UINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### GetAt

- **GetAt**
  - **nOffset**: LINT
  - **nValue**: Reference To UINT
  - **HRESULT GetAt**

  Gets the value

### Syntax

**Definition:**

```c
METHOD GetAt : HRESULT
VAR_INPUT
  nOffset : LINT;
  nValue : Reference To UINT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>nValue</td>
<td>Reference To UINT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

### Return value

**HRESULT [135]**

### SetAt

- **SetAt**
  - **nOffset**: LINT
  - **nValue**: UINT
  - **HRESULT SetAt**

  Sets the value
Syntax

Definition:

```c
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    nValue : UINT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>nValue</td>
<td>UINT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

### 6.3.1.3.33 ITCvNRandomAccess_ULINT

Offers a random access interface for ULINT values.

**Inheritance Hierarchy**

```
ITcUnknown [399]
    ITCvNAccess_ULINT [275]
        ITCvNRandomAccess_ULINT
```

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITCUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITCUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITCUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [276]</td>
<td>ITCvNAccess_ULINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [276]</td>
<td>ITCvNAccess_ULINT</td>
<td>Sets the value</td>
</tr>
<tr>
<td>GetAt [331]</td>
<td>ITCvNRandomAccess_ULINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt [332]</td>
<td>ITCvNRandomAccess_ULINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**GetAt**

```c
nOffset LINT
nValue Reference To ULINT
HRESULT GetAt
```
Gets the value

**Syntax**

Definition:

```c
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    nValue : Reference To ULINT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>nValue</td>
<td>Reference To ULINT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

---

**SetAt**

Sets the value

**Syntax**

Definition:

```c
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    nValue : ULINT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>nValue</td>
<td>ULINT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

---

6.3.1.3.34 ITCvNRandomAccess_USINT

Offers a random access interface for USINT values
Inheritance Hierarchy

ITcUnknown [399]
  ITcVnAccess_USINT [276]
  ITcVnRandomAccess_USINT

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Get [277]</td>
<td>ITcVnAccess_USINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>Set [277]</td>
<td>ITcVnAccess_USINT</td>
<td>Sets the value</td>
</tr>
<tr>
<td>GetAt [333]</td>
<td>ITcVnRandomAccess_USINT</td>
<td>Gets the value</td>
</tr>
<tr>
<td>SetAt [334]</td>
<td>ITcVnRandomAccess_USINT</td>
<td>Sets the value</td>
</tr>
</tbody>
</table>

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

GetAt

```
GetAt

nOffset  LINT
nValue   Reference To USINT
```

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
  nOffset : LINT;
  nValue  : Reference To USINT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>nValue</td>
<td>Reference To USINT</td>
<td>Returns the value</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]
SetAt

Sets the value

Syntax

Definition:

METHOD SetAt : HRESULT
VAR_INPUT
  nOffset : LINT;
  nValue : USINT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset to the current position</td>
</tr>
<tr>
<td>nValue</td>
<td>USINT</td>
<td>The value to set</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.3.1.4 Iterators

This group contains interfaces for the handling of iterators in order to access elements of containers [143].

6.3.1.4.1 ITcVnBidirectionalIterator

Offers an interface for a bidirectional iterator.

Inheritance Hierarchy

ITcUnknown [399]
  ITcVnIteratorBase [338]
    ITcVnForwardIterator [336]
    ITcVnBidirectionalIterator
Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>CheckIfBegin [339]</td>
<td>ITcVnIteratorBase</td>
<td>Checks if the iterator points to the first element.</td>
</tr>
<tr>
<td>CheckIfEnd [339]</td>
<td>ITcVnIteratorBase</td>
<td>Checks if the iterator points to the past-the-end element (alternatively use F_VN_CheckIfIteratorIsAtEnd).</td>
</tr>
<tr>
<td>GetValueSize [340]</td>
<td>ITcVnIteratorBase</td>
<td>Gets the memory size (in byte) required by the value the iterator points to.</td>
</tr>
<tr>
<td>GetValueTypeGuid [340]</td>
<td>ITcVnIteratorBase</td>
<td>Gets the type GUID of the value the iterator points to.</td>
</tr>
<tr>
<td>SetToBegin [341]</td>
<td>ITcVnIteratorBase</td>
<td>Sets the iterator to the first element (alternatively use F_VN_SetIteratorToBegin).</td>
</tr>
<tr>
<td>SetToEnd [341]</td>
<td>ITcVnIteratorBase</td>
<td>Sets the iterator to the past-the-end element.</td>
</tr>
<tr>
<td>CheckIfEqualTo [336]</td>
<td>ITcVnForwardIterator</td>
<td>Checks if iterator is equal to another iterator.</td>
</tr>
<tr>
<td>GetContainer [337]</td>
<td>ITcVnForwardIterator</td>
<td>Gets a pointer to the current element converted into an ITcVnContainer interface and increment its reference counter (only possible for container types). (Alternatively use F_VN_GetContainer.)</td>
</tr>
<tr>
<td>Increment [337]</td>
<td>ITcVnForwardIterator</td>
<td>Increments the iterator. (Alternatively use F_VN_IncrementIterator.)</td>
</tr>
<tr>
<td>SetContainer [338]</td>
<td>ITcVnForwardIterator</td>
<td>Sets the current element using an ITcVnContainer interface (only possible for container types). (Alternatively use F_VN_SetContainer.)</td>
</tr>
<tr>
<td>Decrement [335]</td>
<td>ITcVnBidirectionalIterator</td>
<td>Decrements the iterator.</td>
</tr>
</tbody>
</table>

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC orCX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

Decrement

Decrement

HRESULT Decrement

Decrements the iterator.

Syntax

Definition:

METHOD Decrement : HRESULT

Return value

HRESULT [135]
6.3.1.4.2 ITcVnForwardIterator

Offers an interface for a forward iterator.

Inheritance Hierarchy

ITcUnknown [399]
  ITcVnIteratorBase [338]
  ITcVnForwardIterator

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>CheckIfBegin [339]</td>
<td>ITcVnIteratorBase</td>
<td>Checks if the iterator points to the first element.</td>
</tr>
<tr>
<td>CheckIfEnd [339]</td>
<td>ITcVnIteratorBase</td>
<td>Checks if the iterator points to the past-the-end element (alternatively use F_VN_CheckIfIteratorIsAtEnd).</td>
</tr>
<tr>
<td>GetValueSize [340]</td>
<td>ITcVnIteratorBase</td>
<td>Gets the memory size (in byte) required by the value the iterator points to.</td>
</tr>
<tr>
<td>GetValueTypeGuid [340]</td>
<td>ITcVnIteratorBase</td>
<td>Gets the type GUID of the value the iterator points to.</td>
</tr>
<tr>
<td>SetToBegin [341]</td>
<td>ITcVnIteratorBase</td>
<td>Sets the iterator to the first element (alternatively use F_VN_SetIteratorToBegin).</td>
</tr>
<tr>
<td>SetToEnd [341]</td>
<td>ITcVnIteratorBase</td>
<td>Sets the iterator to the past-the-end element.</td>
</tr>
<tr>
<td>CheckIfEqualTo [336]</td>
<td>ITcVnForwardIterator</td>
<td>Checks if iterator is equal to another iterator.</td>
</tr>
<tr>
<td>GetContainer [337]</td>
<td>ITcVnForwardIterator</td>
<td>Gets a pointer to the current element converted into an ITcVnContainer interface and increment its reference counter (only possible for container types). (Alternatively use F_VN_GetContainer.)</td>
</tr>
<tr>
<td>Increment [337]</td>
<td>ITcVnForwardIterator</td>
<td>Increments the iterator. (Alternatively use F_VN_IncrementIterator.)</td>
</tr>
<tr>
<td>SetContainer [338]</td>
<td>ITcVnForwardIterator</td>
<td>Sets the current element using an ITcVnContainer interface (only possible for container types). (Alternatively use F_VN_SetContainer.)</td>
</tr>
</tbody>
</table>

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

CheckIfEqualTo

CheckIfEqualTo

Checks if iterator is equal to another iterator.

Syntax

Definition:
METHOD CheckIfEqualTo : HRESULT
VAR_INPUT
  ipIterator : ITCvNForwardIterator;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipIterator</td>
<td>ITCvNForwardIterator</td>
<td>Iterator interface to compare with</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

GetContainer

GetContainer

<table>
<thead>
<tr>
<th>pipContainer</th>
<th>Pointer To ITCvNContainer</th>
<th>HRESULT GetContainer</th>
</tr>
</thead>
</table>

Gets a pointer to the current element converted into an ITCvNContainer interface and increment its reference counter (only possible for container types). (Alternatively use F_VN_GetContainer.)

**Syntax**

Definition:

METHOD GetContainer : HRESULT
VAR_INPUT
  pipContainer : Pointer To ITCvNContainer;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pipContainer</td>
<td>Pointer To ITCvNContainer</td>
<td>Returns the container interface.</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

Increment

Increment

<table>
<thead>
<tr>
<th></th>
<th>HRESULT Increment</th>
</tr>
</thead>
</table>

Increments the iterator. (Alternatively use F_VN_IncrementIterator.)

**Syntax**

Definition:

METHOD Increment : HRESULT
SetContainer

Sets the current element using an ITcVnContainer interface (only possible for container types). (Alternatively use F_VN_SetContainer.)

Syntax

**Definition:**

```plaintext
METHOD SetContainer : HRESULT
VAR_INPUT
  ipContainer : ITcVnContainer;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container interface of which the content is to be assigned to the current element.</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

6.3.1.4.3 ITcVnIteratorBase

Offers a base interface for iterators

**Inheritance Hierarchy**

- ITcUnknown [399]
- ITcVnIteratorBase
### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>CheckIfBegin [339]</td>
<td>ITcVnIteratorBase</td>
<td>Checks if the iterator points to the first element.</td>
</tr>
<tr>
<td>CheckIfEnd [339]</td>
<td>ITcVnIteratorBase</td>
<td>Checks if the iterator points to the past-the-end element (alternatively use F_VN_CheckIfIteratorIsAtEnd).</td>
</tr>
<tr>
<td>GetValueSize [340]</td>
<td>ITcVnIteratorBase</td>
<td>Gets the memory size (in byte) required by the value the iterator points to.</td>
</tr>
<tr>
<td>GetValueTypeGuid [340]</td>
<td>ITcVnIteratorBase</td>
<td>Gets the type GUID of the value the iterator points to.</td>
</tr>
<tr>
<td>SetToBegin [341]</td>
<td>ITcVnIteratorBase</td>
<td>Sets the iterator to the first element (alternatively use F_VN_SetIteratorToBegin).</td>
</tr>
<tr>
<td>SetToEnd [341]</td>
<td>ITcVnIteratorBase</td>
<td>Sets the iterator to the past-the-end element.</td>
</tr>
</tbody>
</table>

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

#### CheckIfBegin

**Syntax**

Definition:

```c
METHOD CheckIfBegin : HRESULT
```

**Return value**

`HRESULT [135]`

#### CheckIfEnd

**Syntax**

Definition:

```c
METHOD CheckIfEnd : HRESULT
```
GetValueSize

Gets the memory size (in byte) required by the value the iterator points to.

Syntax

Definition:

METHOD GetValueSize : HRESULT
VAR_INPUT
   nSize : Reference To ULONG;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nSize</td>
<td>Reference To ULONG</td>
<td>Returns the element size in bytes.</td>
</tr>
</tbody>
</table>

GetValueTypeGuid

Gets the type GUID of the value the iterator points to.

Syntax

Definition:

METHOD GetValueTypeGuid : HRESULT
VAR_INPUT
   nTypeGuid : Reference To GUID;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nTypeGuid</td>
<td>Reference To GUID</td>
<td>Returns the type GUID.</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]
SetToBegin

Sets the iterator to the first element (alternatively use F_VN_SetIteratorToBegin).

Syntax

Definition:

METHOD SetToBegin : HRESULT

Return value

HRESULT [135]

SetToEnd

Sets the iterator to the past-the-end element.

Syntax

Definition:

METHOD SetToEnd : HRESULT

Return value

HRESULT [135]

6.3.1.4.4 ITcVnIteratorCopyCreator

Offers an interface providing a method for creating a new iterator to the same position.

Inheritance Hierarchy

ITcUnknown [399]

 ITcVnIteratorCopyCreator

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>Create</td>
<td>ITcVnIteratorCopyCreator</td>
<td>Create a new iterator pointing to the position of the calling iterator.</td>
</tr>
</tbody>
</table>
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

Create

Create a new iterator pointing to the position of the calling iterator.

Syntax

Definition:

```
METHOD Create : HRESULT
VAR_INPUT
   pipIterator : Pointer To ITcVnForwardIterator;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pipIterator</td>
<td>Pointer To ITcVnForwardIterator</td>
<td>Returns the created iterator.</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.3.1.4.5 ITcVnRandomAccessIterator

Offers an interface for a random access iterator.

Inheritance Hierarchy

```
ITcUnknown [399]
  ITcVnIteratorBase [338]
    ITcVnForwardIterator [336]
      ITcVnBidirectionalIterator [334]
    ITcVnRandomAccessIterator
```
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease</td>
<td>ITcUnknown</td>
<td>-Decrements the reference counter.</td>
</tr>
<tr>
<td>CheckIfBegin</td>
<td>ITcVnIteratorBase</td>
<td>Checks if the iterator points to the first element.</td>
</tr>
<tr>
<td>CheckIfEnd</td>
<td>ITcVnIteratorBase</td>
<td>Checks if the iterator points to the past-the-end element (alternatively use F_VN_CheckIfIteratorIsAtEnd).</td>
</tr>
<tr>
<td>GetValueSize</td>
<td>ITcVnIteratorBase</td>
<td>Gets the memory size (in byte) required by the value the iterator points to.</td>
</tr>
<tr>
<td>GetValueTypeGuid</td>
<td>ITcVnIteratorBase</td>
<td>Gets the type GUID of the value the iterator points to.</td>
</tr>
<tr>
<td>SetToBegin</td>
<td>ITcVnIteratorBase</td>
<td>Sets the iterator to the first element (alternatively use F_VN_SetIteratorToBegin).</td>
</tr>
<tr>
<td>SetToEnd</td>
<td>ITcVnIteratorBase</td>
<td>Sets the iterator to the past-the-end element.</td>
</tr>
<tr>
<td>CheckIfEqualTo</td>
<td>ITcVnForwardIterator</td>
<td>Checks if iterator is equal to another iterator.</td>
</tr>
<tr>
<td>GetContainer</td>
<td>ITcVnForwardIterator</td>
<td>Gets a pointer to the current element converted into an ITcVnContainer interface and increment its reference counter (only possible for container types). (Alternatively use F_VN_GetContainer.)</td>
</tr>
<tr>
<td>Increment</td>
<td>ITcVnForwardIterator</td>
<td>Increments the iterator. (Alternatively use F_VN_IncrementIterator.)</td>
</tr>
<tr>
<td>SetContainer</td>
<td>ITcVnForwardIterator</td>
<td>Sets the current element using an ITcVnContainer interface (only possible for container types). (Alternatively use F_VN_SetContainer.)</td>
</tr>
<tr>
<td>Decrement</td>
<td>ITcVnBidirectionalIterator</td>
<td>Decrements the iterator.</td>
</tr>
<tr>
<td>CheckIfGreaterThan</td>
<td>ITcVnRandomAccessIterator</td>
<td>Checks if the iterator is greater than another iterator.</td>
</tr>
<tr>
<td>CheckIfLessThan</td>
<td>ITcVnRandomAccessIterator</td>
<td>Checks if the iterator is less than another iterator.</td>
</tr>
<tr>
<td>GetContainerAt</td>
<td>ITcVnRandomAccessIterator</td>
<td>Gets a pointer to the element at a specific offset from the current element converted into an ITcVnContainer interface and increment its reference counter (only possible for container types). (Alternatively use F_VN_GetAt_ITcVnContainer.)</td>
</tr>
<tr>
<td>SetContainerAt</td>
<td>ITcVnRandomAccessIterator</td>
<td>Sets the element at a specific offset from the current element using an ITcVnContainer interface (only possible for container types). (Alternatively use F_VN_SetAt_ITcVnContainer.)</td>
</tr>
</tbody>
</table>

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### CheckIfGreaterThan

```c
CheckIfGreaterThan

ipIterator        ITcVnRandomAccessIterator

HRESULT CheckIfGreaterThan
```

Checks if the iterator is greater than another iterator.
Syntax

Definition:

METHOD CheckIfGreaterThan : HRESULT
VAR_INPUT
    ipIterator : ITcVnRandomAccessIterator;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipIterator</td>
<td>ITcVnRandomAccessIterator</td>
<td>Iterator interface to compare with.</td>
</tr>
</tbody>
</table>

Return value

HRESULT [ 135 ]

CheckIfLessThan

CheckIfLessThan

<table>
<thead>
<tr>
<th>ipIterator</th>
<th>ITcVnRandomAccessIterator</th>
<th>HRESULT GetContainerAt</th>
</tr>
</thead>
</table>

Checks if the iterator is less than another iterator.

Syntax

Definition:

METHOD CheckIfLessThan : HRESULT
VAR_INPUT
    ipIterator : ITcVnRandomAccessIterator;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipIterator</td>
<td>ITcVnRandomAccessIterator</td>
<td>Iterator interface to compare with.</td>
</tr>
</tbody>
</table>

Return value

HRESULT [ 135 ]

GetContainerAt

GetContainerAt

<table>
<thead>
<tr>
<th>nOffset</th>
<th>LINT</th>
<th>HRESULT GetContainerAt</th>
</tr>
</thead>
<tbody>
<tr>
<td>pipContainer</td>
<td>Pointer To ITcVnContainer</td>
<td></td>
</tr>
</tbody>
</table>

Gets a pointer to the element at a specific offset from the current element converted into an ITcVnContainer interface and increment its reference counter (only possible for container types). (Alternatively use F_VN_GetAt_ITcVnContainer.)
Syntax
Definition:

METHOD GetContainerAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    pipContainer : Pointer To ITcVnContainer;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset from the current element.</td>
</tr>
<tr>
<td>pipContainer</td>
<td>Pointer To ITcVnContainer</td>
<td>Returns the pointer to the container interface.</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

SetContainerAt

SetContainerAt

METHOD SetContainerAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    ipContainer : ITcVnContainer;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset from the current element.</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container interface of which the content is to be assigned to the current element.</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.3.1.5 ITcVnContainer

Offers an interface for an object container.
Inheritance Hierarchy

ITcUnknown \[\text{399}\]

ITcVnContainer

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [\text{399}]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [\text{399}]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [\text{400}]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>CheckIfBasicContainer [\text{347}]</td>
<td>ITcVnContainer</td>
<td>Checks if the container contains only basic (non-container) elements.</td>
</tr>
<tr>
<td>CheckIfEmpty [\text{347}]</td>
<td>ITcVnContainer</td>
<td>Checks if the container is empty. (Alternatively use F_VN_CheckIfEmpty.)</td>
</tr>
<tr>
<td>GetBidirectionalIterator [\text{347}]</td>
<td>ITcVnContainer</td>
<td>Gets an interface pointer to a bidirectional iterator (if this iterator type is supported by the container) and increment its reference counter.</td>
</tr>
<tr>
<td>GetElementNum [\text{348}]</td>
<td>ITcVnContainer</td>
<td>Gets the size (number of elements) of the container. (Alternatively use F_VN_GetNumberOfElements.)</td>
</tr>
<tr>
<td>GetElementSize [\text{348}]</td>
<td>ITcVnContainer</td>
<td>Gets the size (in byte) of each element in the container.</td>
</tr>
<tr>
<td>GetElementTypeGuid [\text{349}]</td>
<td>ITcVnContainer</td>
<td>Gets the GUID of the container elements.</td>
</tr>
<tr>
<td>GetExportSize [\text{349}]</td>
<td>ITcVnContainer</td>
<td>Gets combined size (in byte) of all elements in the container.</td>
</tr>
<tr>
<td>GetForwardIterator [\text{350}]</td>
<td>ITcVnContainer</td>
<td>Gets an interface pointer to a forward iterator (if this iterator type is supported by the container) and increment its reference counter. (Alternatively use F_VN_GetForwardIterator.)</td>
</tr>
<tr>
<td>GetRandomAccessIterator [\text{350}]</td>
<td>ITcVnContainer</td>
<td>Gets an interface pointer to a random access iterator (if this iterator type is supported by the container) and increment its reference counter. (Alternatively use F_VN_GetRandomAccessIterator.)</td>
</tr>
<tr>
<td>GetTypeGuid [\text{351}]</td>
<td>ITcVnContainer</td>
<td>Gets GUID of the container.</td>
</tr>
<tr>
<td>GetTypeName [\text{351}]</td>
<td>ITcVnContainer</td>
<td>Gets the container type name as a string.</td>
</tr>
</tbody>
</table>

Further information

The interface ITcVnContainer is for handling containers [\text{143}]. Observe the notes in the chapter Interface pointers [\text{131}].

- **Overwriting containers**

  Generally, containers with API functions can simply be overwritten. At present, however, overwriting is not possible if the existing container and the container to be written are of different types. This is signaled by the error code 70E (INCOMPATIBLE).

Related functions

- Basic Container Operations [\text{510}]
- Algebraic Container Operations [\text{427}]
- Container statistics [\text{770}]
- Contour analysis [\text{852}]

Version: 1.3
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.3.1.5.1 CheckIfBasicContainer

CheckIfBasicContainer

HRESUT CheckIfBasicContainer

Checks if the container contains only basic (non-container) elements.

Syntax

Definition:

METHOD CheckIfBasicContainer : HRESULT

Return value

HRESULT [135]

6.3.1.5.2 CheckIfEmpty

CheckIfEmpty

HRESUT CheckIfEmpty

Checks if the container is empty. (Alternatively use F_VN_CheckIfEmpty.)

Syntax

Definition:

METHOD CheckIfEmpty : HRESULT

Return value

HRESULT [135]

6.3.1.5.3 GetBidirectionalIterator

GetBidirectionalIterator

pipIterator Pointer To ITcVnBidirectionalIterator HRESUT GetBidirectionalIterator

Gets an interface pointer to a bidirectional iterator (if this iterator type is supported by the container) and increment its reference counter.

Syntax

Definition:

METHOD GetBidirectionalIterator : HRESULT
VAR_INPUT
  pipIterator : Pointer To ITcVnBidirectionalIterator;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pipIterator</td>
<td>Pointer To ITcVnBidirectionalIterator</td>
<td>Returns the iterator interface.</td>
</tr>
</tbody>
</table>

## Return value

**HRESULT** [135]

### 6.3.1.5.4 GetElementNum

**GetElementNum**

```plaintext
nElementNum | Reference To ULINT | HRESULT | GetElementNum
```

Gets the size (number of elements) of the container. (Alternatively use F_VN_GetNumberOfElements.)

**Syntax**

Definition:

```plaintext
METHOD GetElementNum : HRESULT
VAR_INPUT
   nElementNum : Reference To ULINT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nElementNum</td>
<td>Reference To ULINT</td>
<td>Returns the number of elements in the container.</td>
</tr>
</tbody>
</table>

## Return value

**HRESULT** [135]

### 6.3.1.5.5 GetElementSize

**GetElementSize**

```plaintext
nSize | Reference To ULINT | HRESULT | GetElementSize
```

Gets the size (in byte) of each element in the container.

**Syntax**

Definition:

```plaintext
METHOD GetElementSize : HRESULT
VAR_INPUT
   nSize : Reference To ULINT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nSize</td>
<td>Reference To ULINT</td>
<td>Returns the size in byte of a single element in the container.</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

### 6.3.1.5.6 GetElementTypeGuid

**GetElementTypeGuid**

\[ \text{nTypeGuid} : \text{Reference To GUID}; \]

\[ \text{HRESULT \ GetElementTypeGuid} \]

Gets the GUID of the container elements.

**Syntax**

**Definition:**

METHOD GetElementTypeGuid : HRESULT
VAR_INPUT
  nTypeGuid : Reference To GUID;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nTypeGuid</td>
<td>Reference To GUID</td>
<td>Returns the GUID of the container elements.</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

### 6.3.1.5.7 GetExportSize

**GetExportSize**

\[ \text{nExportSize} : \text{Reference To ULINT}; \]

\[ \text{HRESULT \ GetExportSize} \]

Gets combined size (in byte) of all elements in the container.

**Syntax**

**Definition:**

METHOD GetExportSize : HRESULT
VAR_INPUT
  nExportSize : Reference To ULINT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nExportSize</td>
<td>Reference To ULINT</td>
<td>Returns the combined size (in byte) of all elements in the container.</td>
</tr>
</tbody>
</table>
### Return value

HRESULT [135]

### 6.3.1.5.8 GetForwardIterator

**Syntax**

Definition:

```plaintext
METHOD GetForwardIterator : HRESULT
VAR_INPUT
  pipIterator : Pointer To ITCnForwardIterator;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pipIterator</td>
<td>Pointer To ITCnForwardIterator</td>
<td>Returns the iterator interface.</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### 6.3.1.5.9 GetRandomAccessIterator

**Syntax**

Definition:

```plaintext
METHOD GetRandomAccessIterator : HRESULT
VAR_INPUT
  pipIterator : Pointer To ITCnRandomAccessIterator;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pipIterator</td>
<td>Pointer To ITCnRandomAccessIterator</td>
<td>Returns the iterator interface.</td>
</tr>
</tbody>
</table>
**6.3.1.5.10 GetTypeGuid**

**Gets GUID of the container.**

**Syntax**

Definition:

```
METHOD GetTypeGuid : HRESULT
VAR_INPUT
   nTypeGuid : Reference To GUID;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nTypeGuid</td>
<td>Reference To GUID</td>
<td>Returns the GUID of the container.</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

---

**6.3.1.5.11 GetTypeName**

**Gets the container type name as a string.**

**Syntax**

Definition:

```
METHOD GetTypeName : HRESULT
VAR_INPUT
   sTypeName : STRING;
   nMaxLen : UINT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sTypeName</td>
<td>STRING</td>
<td>Returns the container type name as a string.</td>
</tr>
<tr>
<td>nMaxLen</td>
<td>UINT</td>
<td>Maximum string length allowed to be written in sTypeName</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]
6.3.2 Images

This group contains interfaces for recording and handling images [140].

6.3.2.1 Acquisition

This group contains interfaces for handling the image acquisition.

6.3.2.1.1 ITcIoFileImageAcquisition

Interface for file image acquisition.

Inheritance Hierarchy

ITcUnknown [399]
  ITcIoImageAcquisition [356]
  ITcIoFileImageAcquisition

Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>RegisterReceiver [357]</td>
<td>ITcIoImageAcquisition</td>
<td>Register callback interface.</td>
</tr>
<tr>
<td>UnregisterReceiver [358]</td>
<td>ITcIoImageAcquisition</td>
<td>Unregister callback interface.</td>
</tr>
<tr>
<td>OpenCamera [358]</td>
<td>ITcIoImageAcquisition</td>
<td>Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).</td>
</tr>
<tr>
<td>CloseCamera [358]</td>
<td>ITcIoImageAcquisition</td>
<td>Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).</td>
</tr>
<tr>
<td>StartAcquisition [359]</td>
<td>ITcIoImageAcquisition</td>
<td>Start the image acquisition.</td>
</tr>
<tr>
<td>StopAcquisition [359]</td>
<td>ITcIoImageAcquisition</td>
<td>Stop the image acquisition.</td>
</tr>
<tr>
<td>SoftwareTrigger [359]</td>
<td>ITcIoImageAcquisition</td>
<td>Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.</td>
</tr>
<tr>
<td>InitializeCamera [360]</td>
<td>ITcIoImageAcquisition</td>
<td>Send initialization commands to the camera.</td>
</tr>
<tr>
<td>CheckConnection [353]</td>
<td>ITcIoFileImageAcquisition</td>
<td>Checks the connection</td>
</tr>
<tr>
<td>TriggerImage [353]</td>
<td>ITcIoFileImageAcquisition</td>
<td>Initialize the software trigger and trigger a single image. Skips nSkipImages and triggers the capturing of image nSkipImages+1.</td>
</tr>
<tr>
<td>TriggerImageByName [353]</td>
<td>ITcIoFileImageAcquisition</td>
<td>Initialize the software trigger and trigger a single image specified by its name in the client assistant.</td>
</tr>
</tbody>
</table>

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
CheckConnection

**CheckConnection**

```
HRESULT CheckConnection
```

Checks the connection

**Syntax**

**Definition:**

```
METHOD CheckConnection : HRESULT
```

**Return value**

`HRESULT [135]`

TriggerImage

**TriggerImage**

```
nSkipImages DINT HRESULT TriggerImage
```

Initialize the software trigger and trigger a single image. Skips `nSkipImages` and triggers the capturing of image `nSkipImages+1`.

**Syntax**

**Definition:**

```
METHOD TriggerImage : HRESULT
VAR_INPUT
  nSkipImages : DINT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>nSkipImages</code></td>
<td><code>DINT</code></td>
<td>Number of images to skip</td>
</tr>
</tbody>
</table>

**Return value**

`HRESULT [135]`

TriggerImageByName

**TriggerImageByName**

```
sImageName STRING HRESULT TriggerImageByName
```

Initialize the software trigger and trigger a single image specified by its name in the client assistant.

**Syntax**

**Definition:**

```
METHOD TriggerImageByName : HRESULT
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sImageName</code></td>
<td><code>STRING</code></td>
</tr>
</tbody>
</table>

**Return value**

`HRESULT [135]`
METHOD TriggerImageByName : HRESULT
VAR_INPUT
 sImageName : STRING;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sImageName</td>
<td>STRING</td>
<td>Image name to trigger</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.3.2.1.2  ITcIoFileImageRecv

Interface for a image receiver.

Inheritance Hierarchy

ITcUnknown [399]
 ITcIoImageRecv [360]
 ITcIoFileImageRecv

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>ReceiveImage</td>
<td>ITcIoImageRecv</td>
<td>Receive an image from an instance of ITcIoGevImageAcquisition.</td>
</tr>
<tr>
<td>ReceiveOpResult</td>
<td>ITcIoImageRecv</td>
<td>Receive an operation result from an instance of ITcIoGevImageAcquisition.</td>
</tr>
<tr>
<td>ReceiveImage</td>
<td>ITcIoFileImageRecv</td>
<td>Receive an image from an instance of ITcIoFileImageAcquisition.</td>
</tr>
</tbody>
</table>

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

ReceiveImage

Receive an image from an instance of ITcIoFileImageAcquisition.
**Syntax**

**Definition:**

```plaintext
METHOD ReceiveImage : HRESULT
VAR_INPUT
    ipImage : ITcVnImageBase;
    sFileName : STRING;
    hrAcquisitionResult : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage</td>
<td>ITcVnImageBase</td>
<td>Interface pointer to the image.</td>
</tr>
<tr>
<td>sFileName</td>
<td>STRING</td>
<td>Returns the filename to the image.</td>
</tr>
<tr>
<td>hrAcquisitionResult</td>
<td>HRESULT</td>
<td>HRESULT indicating the status of the image acquisition.</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

### 6.3.2.1.3 ITcIoGevImageAcquisition

Interface for GigE Vision image acquisition.

**Inheritance Hierarchy**

ITcUnknown [399]

ITcIoImageAcquisition [356]

ITcIoGevImageAcquisition

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>RegisterReceiver [357]</td>
<td>ITcIoImageAcquisition</td>
<td>Register callback interface.</td>
</tr>
<tr>
<td>UnregisterReceiver [358]</td>
<td>ITcIoImageAcquisition</td>
<td>Unregister callback interface.</td>
</tr>
<tr>
<td>OpenCamera [358]</td>
<td>ITcIoImageAcquisition</td>
<td>Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).</td>
</tr>
<tr>
<td>CloseCamera [358]</td>
<td>ITcIoImageAcquisition</td>
<td>Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).</td>
</tr>
<tr>
<td>StartAcquisition [359]</td>
<td>ITcIoImageAcquisition</td>
<td>Start the image acquisition.</td>
</tr>
<tr>
<td>StopAcquisition [359]</td>
<td>ITcIoImageAcquisition</td>
<td>Stop the image acquisition.</td>
</tr>
<tr>
<td>SoftwareTrigger [359]</td>
<td>ITcIoImageAcquisition</td>
<td>Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.</td>
</tr>
<tr>
<td>InitializeCamera [360]</td>
<td>ITcIoImageAcquisition</td>
<td>Send initialization commands to the camera.</td>
</tr>
<tr>
<td>CheckConnection [356]</td>
<td>ITcIoGevImageAcquisition</td>
<td>Checks the camera connection.</td>
</tr>
</tbody>
</table>
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

CheckConnection

Syntax

Definition:

METHOD CheckConnection : HRESULT
VAR_INPUT
  eAssumedState : Reference To GEV_CAMERA_STATE;
  eActualState : Reference To GEV_CAMERA_STATE;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eAssumedState</td>
<td>Reference To GEV_CAMERA_STATE [203]</td>
<td>The internally assumed state of the camera.</td>
</tr>
<tr>
<td>eActualState</td>
<td>Reference To GEV_CAMERA_STATE [203]</td>
<td>The actually observable state of the camera.</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.3.2.1.4 ITcIoImageAcquisition

Interface for image acquisition.

Inheritance Hierarchy

ITcUnknown [399]
ITcIoImageAcquisition
# Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>RegisterReceiver</td>
<td>ITcIoImageAcquisition</td>
<td>Register callback interface.</td>
</tr>
<tr>
<td>UnregisterReceiver</td>
<td>ITcIoImageAcquisition</td>
<td>Unregister callback interface.</td>
</tr>
<tr>
<td>OpenCamera</td>
<td>ITcIoImageAcquisition</td>
<td>Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).</td>
</tr>
<tr>
<td>CloseCamera</td>
<td>ITcIoImageAcquisition</td>
<td>Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).</td>
</tr>
<tr>
<td>StartAcquisition</td>
<td>ITcIoImageAcquisition</td>
<td>Start the image acquisition.</td>
</tr>
<tr>
<td>StopAcquisition</td>
<td>ITcIoImageAcquisition</td>
<td>Stop the image acquisition.</td>
</tr>
<tr>
<td>SoftwareTrigger</td>
<td>ITcIoImageAcquisition</td>
<td>Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.</td>
</tr>
<tr>
<td>InitializeCamera</td>
<td>ITcIoImageAcquisition</td>
<td>Send initialization commands to the camera.</td>
</tr>
</tbody>
</table>

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## RegisterReceiver

### Syntax

**Definition:**

```plaintext
METHOD RegisterReceiver : HRESULT
VAR_INPUT
    ipRecv : ITcIoImageRecv;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipRecv</td>
<td>ITcIoImageRecv [360]</td>
<td>Pointer to an interface containing the callback function.</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]
UnregisterReceiver

Syntax
 Definition:
 METHOD UnregisterReceiver : HRESULT

Return value
 HRESULT [135]

OpenCamera

Syntax
 Definition:
 METHOD OpenCamera : HRESULT

Return value
 HRESULT [135]

CloseCamera

Syntax
 Definition:
 METHOD CloseCamera : HRESULT

Return value
 HRESULT [135]
StartAcquisition

**StartAcquisition**

```plaintext
HRESULT StartAcquisition
```

Start the image acquisition.

**Syntax**

**Definition:**

```plaintext
METHOD StartAcquisition : HRESULT
```

**Return value**

**HRESULT [135]**

StopAcquisition

**StopAcquisition**

```plaintext
HRESULT StopAcquisition
```

Stop the image acquisition.

**Syntax**

**Definition:**

```plaintext
METHOD StopAcquisition : HRESULT
```

**Return value**

**HRESULT [135]**

SoftwareTrigger

**SoftwareTrigger**

```plaintext
SoftwareTrigger
```

Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.

**Syntax**

**Definition:**

```plaintext
METHOD SoftwareTrigger : HRESULT
VAR_INPUT
   bSplitConcatenatedCommands : BOOL;
   bOmitAcknowledgement : BOOL;
END_VAR
```
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bSplitConcatenatedCommand</td>
<td>BOOL</td>
<td>If true, multi-read and multi-write commands are split into</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sequences of single read and single write commands, respectively.</td>
</tr>
<tr>
<td>bOmitAcknowledgement</td>
<td>BOOL</td>
<td>Indicates that no acknowledge packet should be requested.</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

InitializeCamera

**InitializeCamera**

Send initialization commands to the camera.

Syntax

Definition:

```c
METHOD InitializeCamera : HRESULT
```

Return value

HRESULT [135]

6.3.2.1.5 ITcIoImageRecv

Interface for a image receiver.

Inheritance Hierarchy

ITcUnknown [399]

ITcIoImageRecv

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>ReceiveImage [361]</td>
<td>ITcIoImageRecv</td>
<td>Receive an image from an instance of ITcIoGevImageAcquisition.</td>
</tr>
<tr>
<td>ReceiveOpResult [361]</td>
<td>ITcIoImageRecv</td>
<td>Receive an operation result from an instance of ITcIoGevImageAcquisition.</td>
</tr>
</tbody>
</table>
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**ReceiveImage**

*ReceiveImage*

- `ipImage` : `ITcVnImageBase`
- `hrAcquisitionResult` : `HRESULT`

Receive an image from an instance of ITcIoGevImageAcquisition.

**Syntax**

Definition:

```
METHOD ReceiveImage : HRESULT
VAR_INPUT
   ipImage : ITcVnImageBase;
   hrAcquisitionResult : HRESULT;
END_VAR
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ipImage</code></td>
<td><code>ITcVnImageBase</code></td>
<td>Interface pointer to the image.</td>
</tr>
<tr>
<td><code>hrAcquisitionResult</code></td>
<td><code>HRESULT</code></td>
<td>HRESULT indicating the status of the image acquisition.</td>
</tr>
</tbody>
</table>

**Return value**

`HRESULT` [135]

**ReceiveOpResult**

*ReceiveOpResult*

- `hrOperationResult` : `HRESULT`

Receive an operation result from an instance of ITcIoGevImageAcquisition.

**Syntax**

Definition:

```
METHOD ReceiveOpResult : HRESULT
VAR_INPUT
   hrOperationResult : HRESULT;
END_VAR
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>hrOperationResult</code></td>
<td><code>HRESULT</code></td>
<td>HRESULT indicating the status of the performed operation.</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

6.3.2.1.6 ITCnFileImageProvider

Interface for an image provider for images from outside realtime environment.

Inheritance Hierarchy

ITcUnknown [399]
   ITCnImageProvider [366]
      ITCnFileImageProvider

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITCUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITCUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITCUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>CheckIfBusy [367]</td>
<td>ITCnImageProvider</td>
<td>Check whether the module is busy; i.e., an operation is in progress.</td>
</tr>
<tr>
<td>CheckIfCameraConnected [368]</td>
<td>ITCnImageProvider</td>
<td>Check if the camera is properly connected (Returns S_OK if connected and S_FALSE if not. Can only check software connection, i.e., the check is unable to determine if the cable is connected or not in case that S_FALSE is returned).</td>
</tr>
<tr>
<td>GetCurrentImage [368]</td>
<td>ITCnImageProvider</td>
<td>Gets the current image by detaching its internal reference.</td>
</tr>
<tr>
<td>GetError [368]</td>
<td>ITCnImageProvider</td>
<td>Gets result of the last operation performed.</td>
</tr>
<tr>
<td>OpenCamera [369]</td>
<td>ITCnImageProvider</td>
<td>Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).</td>
</tr>
<tr>
<td>CloseCamera [369]</td>
<td>ITCnImageProvider</td>
<td>Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).</td>
</tr>
<tr>
<td>StartAcquisition [369]</td>
<td>ITCnImageProvider</td>
<td>Start image acquisition.</td>
</tr>
<tr>
<td>StopAcquisition [370]</td>
<td>ITCnImageProvider</td>
<td>Stop image acquisition.</td>
</tr>
<tr>
<td>SoftwareTrigger [370]</td>
<td>ITCnImageProvider</td>
<td>Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.</td>
</tr>
<tr>
<td>InitializeCamera [370]</td>
<td>ITCnImageProvider</td>
<td>Set camera to initial state.</td>
</tr>
<tr>
<td>GetCurrentImageAndFileName [363]</td>
<td>ITCnFileImageProvider</td>
<td>Get the current image by detaching its internal reference.</td>
</tr>
<tr>
<td>TriggerImage [363]</td>
<td>ITCnFileImageProvider</td>
<td>Initialize the software trigger and trigger a single image. Skips nSkipImages and triggers the capturing of image nSkipImages+1.</td>
</tr>
<tr>
<td>TriggerImageByName [364]</td>
<td>ITCnFileImageProvider</td>
<td>Initialize the software trigger and trigger a single image specified by its name in the client assistant.</td>
</tr>
</tbody>
</table>

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

362 Version: 1.3 TF7000 - TF7300
GetCurrentImageAndFileName

Get the current image by detaching its internal reference.

**Syntax**

**Definition:**

```c
METHOD GetCurrentImageAndFileName : HRESULT
VAR_INPUT
   pipImage : Pointer To ITcVnImage;
   sFileName : STRING;
   nMaxLen : UINT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pipImage</td>
<td>Pointer To ITcVnImage</td>
<td>pointer to the interface pointer to be returned, might be 0 if the image acquisition failed</td>
</tr>
<tr>
<td>sFileName</td>
<td>STRING</td>
<td>returns the filename as a string</td>
</tr>
<tr>
<td>nMaxLen</td>
<td>UINT</td>
<td>maximum string length allowed to be written in sFileName</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

TriggerImage

Initialize the software trigger and trigger a single image. Skips nSkipImages and triggers the capturing of image nSkipImages+1.

**Syntax**

**Definition:**

```c
METHOD TriggerImage : HRESULT
VAR_INPUT
   nSkipImages : DINT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nSkipImages</td>
<td>DINT</td>
<td>Number of images to skip</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]
TriggerImageByName

Initialize the software trigger and trigger a single image specified by its name in the client assistant.

Syntax

Definition:

METHOD TriggerImageByName : HRESULT
VAR_INPUT
    sImageName : STRING;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sImageName</td>
<td>STRING</td>
<td>Image name to trigger</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.3.2.1.7 ITcVnGevImageProvider

Interface for a GigE Vision image provider.

Inheritance Hierarchy

ITcUnknown [399]
    ITcVnImageProvider [366]
    ITcVnGevImageProvider


### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITCUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITCUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITCUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>CheckIfBusy [367]</td>
<td>ITcVnImageProvider</td>
<td>Check whether the module is busy; i.e., an operation is in progress.</td>
</tr>
<tr>
<td>CheckIfCameraConnected [368]</td>
<td>ITcVnImageProvider</td>
<td>Check if the camera is properly connected (Returns S_OK if connected and S_FALSE if not. Can only check software connection, i.e. the check is unable to determine if the cable is connected or not in case that S_FALSE is returned).</td>
</tr>
<tr>
<td>GetCurrentImage [368]</td>
<td>ITcVnImageProvider</td>
<td>Gets the current image by detaching its internal reference.</td>
</tr>
<tr>
<td>GetError [368]</td>
<td>ITcVnImageProvider</td>
<td>Gets result of the last operation performed.</td>
</tr>
<tr>
<td>OpenCamera [369]</td>
<td>ITcVnImageProvider</td>
<td>Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).</td>
</tr>
<tr>
<td>CloseCamera [369]</td>
<td>ITcVnImageProvider</td>
<td>Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).</td>
</tr>
<tr>
<td>StartAcquisition [369]</td>
<td>ITcVnImageProvider</td>
<td>Start image acquisition.</td>
</tr>
<tr>
<td>StopAcquisition [370]</td>
<td>ITcVnImageProvider</td>
<td>Stop image acquisition.</td>
</tr>
<tr>
<td>SoftwareTrigger [370]</td>
<td>ITcVnImageProvider</td>
<td>Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.</td>
</tr>
<tr>
<td>InitializeCamera [370]</td>
<td>ITcVnImageProvider</td>
<td>Set camera to initial state.</td>
</tr>
<tr>
<td>GetCurrentGevImage [365]</td>
<td>ITcVnGevImageProvider</td>
<td>Gets the current GigE Vision image by detaching its internal reference.</td>
</tr>
<tr>
<td>GetCurrentImageWithGvspInfo [366]</td>
<td>ITcVnGevImageProvider</td>
<td>Gets the current image by detaching its internal reference and additionally provide the GVSP info.</td>
</tr>
</tbody>
</table>

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### GetCurrentGevImage

```c
pipGevImage : Pointer To ITcIoGevImage
HRESULT GetCurrentGevImage
```

Gets the current GigE Vision image by detaching its internal reference.

### Syntax

**Definition:**

```c
METHOD GetCurrentGevImage : HRESULT
VAR_INPUT
   pipGevImage : Pointer To ITcIoGevImage;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pipGevImage</td>
<td>Pointer To ITcGevImage</td>
<td>Pointer to the interface pointer to be returned, might be 0 if the image acquisition failed.</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## GetCurrentImageWithGvspInfo

```c
METHOD GetCurrentImageWithGvspInfo : HRESULT
VAR_INPUT
    pipImage : Pointer To ITcVnImage;
    stGvspInfo : Reference To GVSP_IMAGE_INFO;
END_VAR
```

Gets the current image by detaching its internal reference and additionally providing the GVSP info.

### Syntax

**Definition:**

```c
METHOD GetCurrentImageWithGvspInfo : HRESULT
VAR_INPUT
    pipImage : Pointer To ITcVnImage;
    stGvspInfo : Reference To GVSP_IMAGE_INFO;
END_VAR
```

## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pipImage</td>
<td>Pointer To ITcVnImage</td>
<td>Pointer to the interface pointer to be returned, might be 0 if the image acquisition failed.</td>
</tr>
<tr>
<td>stGvspInfo</td>
<td>Reference To GVSP_IMAGE_INFO</td>
<td>Contains useful meta information like image id, time stamp etc.</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

### 6.3.2.1.8 ITcVnImageProvider

Interface for an image provider.

**Inheritance Hierarchy**

ITcUnknown [399]
ITcVnImageProvider
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>CheckIfBusy [367]</td>
<td>ITcVnImageProvider</td>
<td>Check whether the module is busy; i.e, an operation is in progress.</td>
</tr>
<tr>
<td>CheckIfCameraConnected [368]</td>
<td>ITcVnImageProvider</td>
<td>Check if the camera is properly connected (Returns S_OK if connected and S_FALSE if not. Can only check software connection, i.e. the check is unable to determine if the cable is connected or not in case that S_FALSE is returned).</td>
</tr>
<tr>
<td>GetCurrentImage [368]</td>
<td>ITcVnImageProvider</td>
<td>Gets the current image by detaching its internal reference.</td>
</tr>
<tr>
<td>GetError [368]</td>
<td>ITcVnImageProvider</td>
<td>Gets result of the last operation performed.</td>
</tr>
<tr>
<td>OpenCamera [369]</td>
<td>ITcVnImageProvider</td>
<td>Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).</td>
</tr>
<tr>
<td>CloseCamera [369]</td>
<td>ITcVnImageProvider</td>
<td>Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).</td>
</tr>
<tr>
<td>StartAcquisition [369]</td>
<td>ITcVnImageProvider</td>
<td>Start image acquisition.</td>
</tr>
<tr>
<td>StopAcquisition [370]</td>
<td>ITcVnImageProvider</td>
<td>Stop image acquisition.</td>
</tr>
<tr>
<td>SoftwareTrigger [370]</td>
<td>ITcVnImageProvider</td>
<td>Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.</td>
</tr>
<tr>
<td>InitializeCamera [370]</td>
<td>ITcVnImageProvider</td>
<td>Set camera to initial state.</td>
</tr>
</tbody>
</table>

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## CheckIfBusy

**Syntax**

**Definition:**

```c
METHOD CheckIfBusy : HRESULT
```

**Return value**

**HRESULT [135]**
CheckIfCameraConnected

Check if the camera is properly connected (Returns S_OK if connected and S_FALSE if not. Can only check software connection, i.e. the check is unable to determine if the cable is connected or not in case that S_FALSE is returned).

Syntax

Definition:

METHOD CheckIfCameraConnected : HRESULT

Return value

HRESULT [135]

GetCurrentImage

Gets the current image by detaching its internal reference.

Syntax

Definition:

METHOD GetCurrentImage : HRESULT

VAR_INPUT

   pipImage : Pointer To ITCvNImage;

END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pipImage</td>
<td>Pointer To ITCvNImage</td>
<td>Pointer to the interface pointer to be returned, might be 0 if the image acquisition failed.</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

GetError

Gets result of the last operation performed.

Syntax

Definition:
METHOD GetError : HRESULT

Return value
HRESULT [135]

OpenCamera

Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).

Syntax
Definition:
METHOD OpenCamera : HRESULT

Return value
HRESULT [135]

CloseCamera

Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).

Syntax
Definition:
METHOD CloseCamera : HRESULT

Return value
HRESULT [135]

StartAcquisition

Start image acquisition.

Syntax
Definition:
METHOD StartAcquisition : HRESULT
StopAcquisition
StopAcquisition
Stop image acquisition.

SoftwareTrigger
SoftwareTrigger
Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.

InitializeCamera
InitializeCamera
Set camera to initial state.
6.3.2.2 Export

This group contains interfaces for handling the data export in connection with images.

6.3.2.2.1 ITcVnBitmapExport

Interface for exporting images as Windows bitmap.

Inheritance Hierarchy

ITcUnknown [399]
   ITcVnBitmapExport

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>GetBitmapSize</td>
<td>ITcVnBitmapExport</td>
<td>Gets size of the image if it is converted to a Windows bitmap.</td>
</tr>
<tr>
<td>GetBitmapImage</td>
<td>ITcVnBitmapExport</td>
<td>Serialize image as Windows bitmap into a given buffer.</td>
</tr>
</tbody>
</table>

Further information

By definition, a bitmap can have only 8 bits (for 1-channel images) or 24 bits (for 3 and 4-channel images). Therefore, 16-bit images as well as images with the element type REAL or LREAL are converted for the bitmap export. In the case of 16-bit images the value range is simply scaled down to 8-bit (0 corresponds to 0 and 255 corresponds to 65,280). In the case of (L)REAL images the value range [-1, +1] is scaled to [-127, +127] and transferred as a signed integer. Values outside these limits will be interpreted as -1 or +1 respectively.

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

GetBitmapSize

GetBitmapSize

<table>
<thead>
<tr>
<th>nBitmapSize</th>
<th>Reference To ULIINT</th>
<th>HRESULT GetBitmapSize</th>
</tr>
</thead>
<tbody>
<tr>
<td>nBitmapWidth</td>
<td>Reference To UDINT</td>
<td></td>
</tr>
<tr>
<td>nBitmapHeight</td>
<td>Reference To UDINT</td>
<td></td>
</tr>
</tbody>
</table>

Gets size of the image if it is converted to a Windows bitmap.

Syntax

Definition:

METHOD GetBitmapSize : HRESULT
VAR_INPUT
nBitmapSize : Reference To ULIINT;
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nBitmapSize</td>
<td>Reference To ULINT</td>
<td>Output parameter containing the required buffer size.</td>
</tr>
<tr>
<td>pDestBuffer</td>
<td>PVOID</td>
<td>Pointer to the destination buffer.</td>
</tr>
<tr>
<td>nBitmapWidth</td>
<td>Reference To UDINT</td>
<td>Desired width or 0 to keep the original width (in) and actual width (out).</td>
</tr>
<tr>
<td>nBitmapHeight</td>
<td>Reference To UDINT</td>
<td>Desired height or 0 to keep the original height (in) and actual height (out).</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### GetBitmapImage

Serialize image as Windows bitmap into a given buffer.

**Syntax**

**Definition:**

```
METHOD GetBitmapImage : HRESULT
VAR_INPUT
  nBitmapSize   : Reference To ULINT;
  pDestBuffer   : PVOID;
  nBitmapWidth  : Reference To UDINT;
  nBitmapHeight : Reference To UDINT;
END_VAR
```

### 6.3.2.2 ITcVnBitmapExportNotification

Interface required for sending displayable images as ADS notifications.
Inheritance Hierarchy

ITcUnknown [399]
   ITcVnBitmapExportNotification

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>GetTimestamp [373]</td>
<td>ITcVnBitmapExportNotification</td>
<td>Gets the timestamp of the latest image change</td>
</tr>
<tr>
<td>GetBitmapImageRpcUnlocked [374]</td>
<td>ITcVnBitmapExportNotification</td>
<td>Serialize the image as Windows bitmap into a given buffer by means of an unlocked remote procedure call. It lies within the responsibility of the user to ensure that no conflicting accesses can occur.</td>
</tr>
</tbody>
</table>

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64)</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### GetTimestamp

```plaintext
GetTimestamp

nTimestamp Reference To LINT HRESULT GetTimestamp
```

Gets the timestamp of the latest image change.

### Syntax

#### Definition:

```plaintext
METHOD GetTimestamp : HRESULT
VAR_INPUT
   nTimestamp : Reference To LINT;
END_VAR
```

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nTimestamp</td>
<td>Reference To LINT</td>
<td>Returns the timestamp</td>
</tr>
</tbody>
</table>

#### Return value

HRESULT [135]
GetBitmapImageRpcUnlocked

Syntax

Definition:

METHOD GetBitmapImageRpcUnlocked : HRESULT
VAR_INPUT
   nBitmapSize : Reference To ULINT;
   pDestBuffer : PVOID;
   nBitmapWidth : Reference To UDINT;
   nBitmapHeight : Reference To UDINT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nBitmapSize</td>
<td>Reference To ULINT</td>
<td>Maximum buffer size (in) and actual buffer size (out).</td>
</tr>
<tr>
<td>pDestBuffer</td>
<td>PVOID</td>
<td>Pointer to the destination buffer.</td>
</tr>
<tr>
<td>nBitmapWidth</td>
<td>Reference To UDINT</td>
<td>Desired width or 0 to keep the original width (in) and actual width (out).</td>
</tr>
<tr>
<td>nBitmapHeight</td>
<td>Reference To UDINT</td>
<td>Desired height or 0 to keep the original height (in) and actual height (out)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.3.2.2.3 ITcVnBitmapExportRpcUnlocked

Interface to serialize an image as Windows bitmap into a given buffer by means of an unlocked remote procedure call.

Inheritance Hierarchy

ITcUnknown [399]
   ITcVnBitmapExport [371]
      ITcVnBitmapExportRpcUnlocked
Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>GetBitmapSize</td>
<td>ITcVnBitmapExport</td>
<td>Gets size of the image if it is converted to a Windows bitmap.</td>
</tr>
<tr>
<td>GetBitmapImage</td>
<td>ITcVnBitmapExport</td>
<td>Serialize image as Windows bitmap into a given buffer.</td>
</tr>
<tr>
<td>GetBitmapImageRpcUnlocked</td>
<td>ITcVnBitmapExportRpcUnlocked</td>
<td>Serialize the image as Windows bitmap into a given buffer by means of an unlocked remote procedure call. It lies within the responsibility of the user to ensure that no conflicting accesses can occur.</td>
</tr>
</tbody>
</table>

Further information

By definition, a bitmap can have only 8 bits (for 1-channel images) or 24 bits (for 3 and 4-channel images). Therefore, 16-bit images as well as images with the element type REAL or LREAL are converted for the bitmap export. In the case of 16-bit images the value range is simply scaled down to 8-bit (0 corresponds to 0 and 255 corresponds to 65,280). In the case of (L)REAL images the value range [-1, +1] is scaled to [-127, +127] and transferred as a signed integer. Values outside these limits will be interpreted as -1 or +1 respectively.

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

GetBitmapImageRpcUnlocked

```plaintext
METHOD GetBitmapImageRpcUnlocked : HRESULT
VAR_INPUT
  nBitmapSize: Reference To ULINT;
  pDestBuffer: PVOID;
  nBitmapWidth: Reference To UDINT;
  nBitmapHeight: Reference To UDINT;
END_VAR
```

Serialize the image as Windows bitmap into a given buffer by means of an unlocked remote procedure call. It lies within the responsibility of the user to ensure that no conflicting accesses can occur.

Syntax

Definition:

```plaintext
METHOD GetBitmapImageRpcUnlocked : HRESULT
VAR_INPUT
  nBitmapSize: Reference To ULINT;
  pDestBuffer: PVOID;
  n(BitmapWidth: Reference To UDINT;
  n(BitmapHeight: Reference To UDINT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nBitmapSize</td>
<td>Reference To ULINT</td>
<td>Maximum buffer size (in) and actual buffer size (out).</td>
</tr>
<tr>
<td>pDestBuffer</td>
<td>PVOID</td>
<td>Pointer to the destination buffer.</td>
</tr>
<tr>
<td>nBitmapWidth</td>
<td>Reference To UDINT</td>
<td>Desired width or 0 to keep the original width (in) and actual width (out).</td>
</tr>
<tr>
<td>nBitmapHeight</td>
<td>Reference To UDINT</td>
<td>Desired height or 0 to keep the original height (in) and actual height (out).</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

### 6.3.2.2.4 ITcVnHistogramExport

Interface for exporting a histogram for an image as an array.

**Inheritance Hierarchy**

ITcUnknown [399]

ITcVnHistogramExport

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>GetHistogramSize [376]</td>
<td>ITcVnHistogramExport</td>
<td>Gets the size of the histogram array.</td>
</tr>
<tr>
<td>GetHistogramArray [377]</td>
<td>ITcVnHistogramExport</td>
<td>Gets the histogram array.</td>
</tr>
<tr>
<td>GetCustomHistogramSize [377]</td>
<td>ITcVnHistogramExport</td>
<td>Gets the size of a custom histogram array.</td>
</tr>
<tr>
<td>GetCustomHistogramArray [378]</td>
<td>ITcVnHistogramExport</td>
<td>Gets a custom histogram array.</td>
</tr>
</tbody>
</table>

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**GetHistogramSize**

GetHistogramSize

nHistogramSize Reference To ULINT HRESULT GetHistogramSize

Gets the size of the histogram array.
## Syntax

**Definition:**

METHOD GetHistogramSize : HRESULT
VAR_INPUT
   nHistogramSize : Reference To ULINT;
END_VAR

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nHistogramSize</td>
<td>Reference To ULINT</td>
<td>Output parameter containing the required buffer size.</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

## GetHistogramArray

### Syntax

**Definition:**

METHOD GetHistogramArray : HRESULT
VAR_INPUT
   nHistogramSize : Reference To ULINT;
   pDestBuffer   : PVOID;
END_VAR

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nHistogramSize</td>
<td>Reference To ULINT</td>
<td>Maximum buffer size (in) and actual buffer size (out).</td>
</tr>
<tr>
<td>pDestBuffer</td>
<td>PVOID</td>
<td>Pointer to the destination buffer.</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

## GetCustomHistogramSize

### Syntax

**Definition:**

METHOD GetCustomHistogramSize : HRESULT
VAR_INPUT
   nHistogramSize : Reference To ULINT;
   nBins         : Reference To UDINT;
END_VAR

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nHistogramSize</td>
<td>Reference To ULINT</td>
<td></td>
</tr>
<tr>
<td>nBins</td>
<td>Reference To UDINT</td>
<td></td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

## Gets the size of a custom histogram array.
Syntax

Definition:

METHOD GetCustomHistogramSize : HRESULT
VAR_INPUT
nHistogramSize : Reference To ULINT;
nBins : Reference To UDINT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nHistogramSize</td>
<td>Reference To ULINT</td>
<td>Output parameter containing the required buffer size.</td>
</tr>
<tr>
<td>nBins</td>
<td>Reference To UDINT</td>
<td>Desired number of bins or 0 to keep the default for the corresponding image format (in) and default number of bins (out).</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

GetCustomHistogramArray

Gets a custom histogram array.

Syntax

Definition:

METHOD GetCustomHistogramArray : HRESULT
VAR_INPUT
nHistogramSize : Reference To ULINT;
pDestBuffer : PVOID;
nBins : Reference To UDINT;
fLowerBound : Reference To LREAL;
fUpperBound : Reference To LREAL;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nHistogramSize</td>
<td>Reference To ULINT</td>
<td>Maximum buffer size (in) and actual buffer size (out).</td>
</tr>
<tr>
<td>pDestBuffer</td>
<td>PVOID</td>
<td>Pointer to the destination buffer.</td>
</tr>
<tr>
<td>nBins</td>
<td>Reference To UDINT</td>
<td>Desired number of bins or 0 to keep the default for the corresponding image format (in) and default number of bins (out).</td>
</tr>
<tr>
<td>fLowerBound</td>
<td>Reference To LREAL</td>
<td>Lower (inclusive) boundary of the 0-th histogram bin (in), or receive the default if fLowerBound AND fUpperBound are set to 0 (out).</td>
</tr>
<tr>
<td>fUpperBound</td>
<td>Reference To LREAL</td>
<td>Upper (exclusive) boundary of the last histogram bin nBins-1 (in), or receive the default if fLowerBound AND fUpperBound are set to 0 (out).</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

### 6.3.2.3 ITcIoGevImage

Interface for GigE Vision images.

### Inheritance Hierarchy

ITcUnknown [399]

ITcVnImageBase [386]

ITcIoGevImage
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>GetImageData [387]</td>
<td>ITcVnImageBase</td>
<td>Gets the data interface (Do not explicitly free the pointer in there!).</td>
</tr>
<tr>
<td>GetImageSize [387]</td>
<td>ITcVnImageBase</td>
<td>Gets the image size.</td>
</tr>
<tr>
<td>GetWidth [388]</td>
<td>ITcVnImageBase</td>
<td>Gets the image width (alternatively use F_VN_GetImageWidth).</td>
</tr>
<tr>
<td>GetHeight [388]</td>
<td>ITcVnImageBase</td>
<td>Gets the image height (alternatively use F_VN_GetImageHeight).</td>
</tr>
<tr>
<td>GetXPadding [389]</td>
<td>ITcVnImageBase</td>
<td>Gets the horizontal padding.</td>
</tr>
<tr>
<td>GetYPadding [389]</td>
<td>ITcVnImageBase</td>
<td>Gets the vertical padding.</td>
</tr>
<tr>
<td>GetImageInfo [390]</td>
<td>ITcVnImageBase</td>
<td>Gets a struct containing all common meta infos of the image. This basically encompasses all meta information accessible via this interface. (Alternatively use F_VN_GetImageInfo.)</td>
</tr>
<tr>
<td>CheckIfCopy [380]</td>
<td>ITcloGevImage</td>
<td>Check if image data is a copy.</td>
</tr>
<tr>
<td>DecreaseHeight [381]</td>
<td>ITcloGevImage</td>
<td>Sets image height to newHeight if it is smaller than the current value.</td>
</tr>
<tr>
<td>GetBlockId [381]</td>
<td>ITcloGevImage</td>
<td>Gets the GVSP block ID.</td>
</tr>
<tr>
<td>GetGevStatus [381]</td>
<td>ITcloGevImage</td>
<td>Gets the block status.</td>
</tr>
<tr>
<td>GetGvspImageInfo [382]</td>
<td>ITcloGevImage</td>
<td>Gets a pointer to the GVSP leader payload.</td>
</tr>
<tr>
<td>SetGevStatus [382]</td>
<td>ITcloGevImage</td>
<td>Sets the block status.</td>
</tr>
</tbody>
</table>

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.3.2.3.1 CheckIfCopy

**CheckIfCopy**

```plaintext
HRESULT CheckIfCopy
```

Check if image data is a copy.

#### Syntax

**Definition:**

```plaintext
METHOD CheckIfCopy : HRESULT
```

#### Return value

**HRESULT [135]**
6.3.2.3.2 DecreaseHeight

Sets image height to newHeight if it is smaller than the current value.

Syntax

**Definition:**

```plaintext
METHOD DecreaseHeight : HRESULT
  VAR_INPUT
  nNewHeight : UDINT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nNewHeight</td>
<td>UDINT</td>
<td>Height value to set.</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

6.3.2.3.3 GetBlockId

Gets the GVSP block ID.

Syntax

**Definition:**

```plaintext
METHOD GetBlockId : HRESULT
  VAR_INPUT
  nBlockId : Reference To ULINT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nBlockId</td>
<td>Reference To ULINT</td>
<td>Returns the block ID.</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

6.3.2.3.4 GetGevStatus

**Definition:**

```plaintext
METHOD GetGevStatus : HRESULT
  VAR_INPUT
  nGevStatus : Reference To UINT;
END_VAR
```
Gets the block status.

**Syntax**

Definition:

```
METHOD GetGevStatus : HRESULT
VAR_INPUT
    nGevStatus : Reference To UINT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nGevStatus</td>
<td>Reference To UINT</td>
<td>Returns the GigE Vision status code.</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

### 6.3.2.3.5 GetGvspImageInfo

**GetGvspImageInfo**

```
ppGvspImageInfo : Pointer To Pointer To GVSP_IMAGE_INFO HRESULT GetGvspImageInfo
```

Gets a pointer to the GVSP leader payload.

**Syntax**

Definition:

```
METHOD GetGvspImageInfo : HRESULT
VAR_INPUT
    ppGvpImageInfo : Pointer To Pointer To GVSP_IMAGE_INFO;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ppGvpImageInfo</td>
<td>Pointer To Pointer To GVSP_IMAGE_INFO</td>
<td>Pointer to the GVSP meta information.</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

### 6.3.2.3.6 SetGevStatus

**SetGevStatus**

```
nGevStatus : UINT HRESULT SetGevStatus
```

Sets the block status.
Syntax

Definition:

METHOD SetGevStatus : HRESULT
VAR_INPUT
    nGevStatus : UINT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nGevStatus</td>
<td>UINT</td>
<td>GigE Vision status code.</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.3.2.4 ITcVnDisplayableImage

Interface for displayable images.

Inheritance Hierarchy

ITcUnknown [399]
    ITcVnDisplayableImage

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
</tbody>
</table>

Further information

This interface is used to display images via ADS (e.g. with ADS Image Watch). It is not suitable for processing images. The ITcVnImage [383] interface is required to process images. For more information, see Distinction between processable and displayable images [140].

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.3.2.5 ITcVnImage

Basic interface for images.

Inheritance Hierarchy

ITcUnknown [399]
    ITcVnImageBase [386]
    ITcVnImage
### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>getImageData</td>
<td>ITcVnImageBase</td>
<td>Gets the data interface (Do not explicitly free the pointer in there!).</td>
</tr>
<tr>
<td>getImageSize</td>
<td>ITcVnImageBase</td>
<td>Gets the image size.</td>
</tr>
<tr>
<td>getWidth</td>
<td>ITcVnImageBase</td>
<td>Gets the image width (alternatively use F_VN_GetImageWidth).</td>
</tr>
<tr>
<td>getHeight</td>
<td>ITcVnImageBase</td>
<td>Gets the image height (alternatively use F_VN_GetImageHeight).</td>
</tr>
<tr>
<td>GetXPadding</td>
<td>ITcVnImageBase</td>
<td>Gets the horizontal padding.</td>
</tr>
<tr>
<td>GetYPadding</td>
<td>ITcVnImageBase</td>
<td>Gets the vertical padding.</td>
</tr>
<tr>
<td>GetPixelFormat</td>
<td>ITcVnImageBase</td>
<td>Gets the pixel format (alternatively use F_VN_GetPixelFormat).</td>
</tr>
<tr>
<td>GetImageInfo</td>
<td>ITcVnImageBase</td>
<td>Gets a struct containing all common meta infos of the image. This basically encompasses all meta information accessible via this interface. (Alternatively use F_VN_GetImageInfo.)</td>
</tr>
<tr>
<td>Init</td>
<td>ITcVnImage</td>
<td>Initialize an image with an ITcVnImageBase interface.</td>
</tr>
<tr>
<td>GetRowPointer</td>
<td>ITcVnImage</td>
<td>Gets a pointer to a specific row of an image.</td>
</tr>
<tr>
<td>ReleaseRowPointer</td>
<td>ITcVnImage</td>
<td>Release the pointer to a specific row of an image.</td>
</tr>
</tbody>
</table>

#### Further information

The interface ITcVnImage is for handling images. It provides methods for obtaining metadata such as image size, number of channels, pixel type etc. and for accessing the raw image data.

This interface is not suitable for image display via ADS (e.g. with ADS Image Watch). The ITcVnDisplayableImage interface is required for image display via ADS. A conversion can be carried out with the following functions:

- F_VN_CopyIntoDisplayableImage
- F_VN_TransformIntoDisplayableImage

#### Related functions

- Basic Image Operations
- Algebraic Image Operations

#### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

#### 6.3.2.5.1 Init

```c
HRESULT Init(ipImageBase ITcVnImageBase Init)
```
Initialize an image with an ITcVnImageBase interface.

Syntax

Definition:

METHOD Init : HRESULT
VAR_INPUT
   ipImageBase : ITcVnImageBase;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImageBase</td>
<td>ITcVnImageBase [386]</td>
<td>Image from which to obtain the data.</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.3.2.5.2 GetRowPointer

Gets a pointer to a specific row of an image.

Syntax

Definition:

METHOD GetRowPointer : HRESULT
VAR_INPUT
   nRowIndex : UDINT;
   ppRow     : Pointer To PVOID;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nRowIndex</td>
<td>UDINT</td>
<td>Row index.</td>
</tr>
<tr>
<td>ppRow</td>
<td>Pointer To PVOID</td>
<td>Returns a pointer to the requested image row.</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Further information

Make sure that the describing pointer is of the same elementary type as the pixel type of the image.

NOTE

Enabling an Row Pointer

If you create a pointer with GetRowPointer, you must also release it again with the method ReleaseRowPointer!
6.3.2.5.3 ReleaseRowPointer

ReleaseRowPointer

```
METHOD ReleaseRowPointer : HRESULT
  VAR_INPUT
    ppRow : Pointer To PVOID;
  END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ppRow</td>
<td>Pointer To PVOID</td>
<td>Pointer to the row pointer to be released.</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

6.3.2.6 ITcVnImageBase

Base interface for all image types.

**Inheritance Hierarchy**

ITcUnknown [399]

ITcVnImageBase

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>GetImageData [387]</td>
<td>ITcVnImageBase</td>
<td>Gets the data interface (Do not explicitly free the pointer in there!).</td>
</tr>
<tr>
<td>GetImageSize [387]</td>
<td>ITcVnImageBase</td>
<td>Gets the image size.</td>
</tr>
<tr>
<td>GetWidth [388]</td>
<td>ITcVnImageBase</td>
<td>Gets the image width (alternatively use F_VN_GetImageWidth).</td>
</tr>
<tr>
<td>GetHeight [388]</td>
<td>ITcVnImageBase</td>
<td>Gets the image height (alternatively use F_VN_GetImageHeight).</td>
</tr>
<tr>
<td>GetXPadding [389]</td>
<td>ITcVnImageBase</td>
<td>Gets the horizontal padding.</td>
</tr>
<tr>
<td>GetYPadding [389]</td>
<td>ITcVnImageBase</td>
<td>Gets the vertical padding.</td>
</tr>
<tr>
<td>GetImageInfo [390]</td>
<td>ITcVnImageBase</td>
<td>Gets a struct containing all common meta infos of the image. This basically encompasses all meta information accessible via this interface. (Alternatively use F_VN_GetImageInfo.)</td>
</tr>
</tbody>
</table>
Further information

The interface ITCvNImageBase is used for internal purposes and is not normally needed by the user.

For handling images use the ITCvNImage interface instead.

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.3.2.6.1 GetImageData

`GetImageData`

```
GetImageData
```

Inputs

- **Name**: pipImageData
  - **Type**: Pointer To ITCvNData
  - **Description**: Returns a pointer to the image data interface.

Return value

```
HRESULT `GetImageData`
```

6.3.2.6.2 GetImageSize

`GetImageSize`

```
GetImageSize
```

Inputs

- **Name**: nSize
  - **Type**: Reference To ULINT
  - **Description**: Returns the image size.

Return value

```
HRESULT `GetImageSize`
```

Syntax

Definition:

```
METHOD GetImageData : HRESULT
VAR_INPUT
   pipImageData : Pointer To ITCvNData;
END_VAR
```

Definition:

```
METHOD GetImageSize : HRESULT
VAR_INPUT
   nSize : Reference To ULINT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nSize</td>
<td>Reference To ULI NT</td>
<td>Returns the image size in bytes.</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

### 6.3.2.6.3 GetWidth

**GetWidth**

<table>
<thead>
<tr>
<th>nWidth</th>
<th>Reference To UDINT</th>
<th>HRESULT GetWidth</th>
</tr>
</thead>
</table>

Gets the image width (alternatively use F_VN_GetImageWidth).

**Syntax**

Definition:

```c
METHOD GetWidth : HRESULT
VAR_INPUT
    nWidth : Reference To UDINT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nWidth</td>
<td>Reference To UDINT</td>
<td>Returns the image width in pixels.</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

### 6.3.2.6.4 GetHeight

**GetHeight**

<table>
<thead>
<tr>
<th>nHeight</th>
<th>Reference To UDINT</th>
<th>HRESULT GetHeight</th>
</tr>
</thead>
</table>

Gets the image height (alternatively use F_VN_GetImageHeight).

**Syntax**

Definition:

```c
METHOD GetHeight : HRESULT
VAR_INPUT
    nHeight : Reference To UDINT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nHeight</td>
<td>Reference To UDINT</td>
<td>Returns the image height in pixels.</td>
</tr>
</tbody>
</table>
6.3.2.6.5  GetXPadding

### Syntax

**Definition:**

```plaintext
METHOD GetXPadding : HRESULT
VAR_INPUT
   nXPadding : Reference To UINT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nXPadding</td>
<td>Reference To UINT</td>
<td>Returns the horizontal padding in bytes.</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

6.3.2.6.6  GetYPadding

### Syntax

**Definition:**

```plaintext
METHOD GetYPadding : HRESULT
VAR_INPUT
   nYPadding : Reference To UINT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nYPadding</td>
<td>Reference To UINT</td>
<td>Returns the vertical padding in bytes.</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]
6.3.2.6.7 GetPixelFormat

**GetPixelFormat**

```
stPixelFormat Reference To TcVnPixelFormat HRESULT GetPixelFormat
```

Gets the pixel format (alternatively use F_VN_GetPixelFormat).

**Syntax**

Definition:

```plaintext
METHOD GetPixelFormat : HRESULT
VAR_INPUT
    stPixelFormat : Reference To TcVnPixelFormat;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stPixelFormat</td>
<td>Reference To TcVnPixelFormat</td>
<td>Returns a struct describing the pixel format.</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

6.3.2.6.8 GetImageInfo

**GetImageInfo**

```
stImageInfo Reference To TcVnImageInfo HRESULT GetImageInfo
```

Gets a struct containing all common meta infos of the image. This basically encompasses all meta information accessible via this interface. (Alternatively use F_VN_GetImageInfo.)

**Syntax**

Definition:

```plaintext
METHOD GetImageInfo : HRESULT
VAR_INPUT
    stImageInfo : Reference To TcVnImageInfo;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stImageInfo</td>
<td>Reference To TcVnImageInfo</td>
<td>Returns a struct describing the image.</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

6.3.3 Machine Learning

This group contains interfaces for machine learning.
6.3.3.1 ITcVnColorModel

Interface to train and classify an image color.

Inheritance Hierarchy

ITcUnknown [399]
  ITcVnMlModel [392]
  ITcVnColorModel

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>GetTypeGuid [393]</td>
<td>ITcVnMlModel</td>
<td>Gets the GUID of the model.</td>
</tr>
<tr>
<td>GetTypeName [394]</td>
<td>ITcVnMlModel</td>
<td>Gets the model type name as a string.</td>
</tr>
<tr>
<td>TrainImageColor [391]</td>
<td>ITcVnColorModel</td>
<td>Train the color of the provided image.</td>
</tr>
<tr>
<td>ClassifyImageColor [392]</td>
<td>ITcVnColorModel</td>
<td>Compute the similarity of each image pixel to the trained reference color.</td>
</tr>
</tbody>
</table>

Further information

The interface ITcVnColorModel is for describing a color area. The model can be created or adjusted with the function F_VN_TrainImageColor [1102] and compared with the colors of an image using the function F_VN_ReferenceColorSimilarity_ITcVnColorModel [1097].

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.3.3.1.1 TrainImageColor

TrainImageColor

deprecated

Train the color of the provided image.

Syntax

Definition:

METHOD TrainImageColor : HRESULT
VAR_INPUT
  - ipSrcImage : ITcVnImage;
  - nDifferentColors : UDINT;
  - ipMask : ITcVnImage;
  - nSkipPixels : UDINT;
END_VAR
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image.</td>
</tr>
<tr>
<td>nDifferentColors</td>
<td>UDINT</td>
<td>Number of different colors to differentiate.</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITcVnImage</td>
<td>Optional image mask.</td>
</tr>
<tr>
<td>nSkipPixels</td>
<td>UDINT</td>
<td>Number of pixels to skip between each evaluated color sample (to achieve a better performance). 0 takes every pixel into account and tends to be more accurate.</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### 6.3.3.1.2 ClassifyImageColor

**Syntax**

```
METHOD ClassifyImageColor : HRESULT
VAR_INPUT
    ipSrcImage     : ITcVnImage;
    pipDestImage   : Pointer To ITcVnImage;
    fVariance      : REAL;
    fLuminanceWeight : REAL;
END_VAR
```

Compute the similarity of each image pixel to the trained reference color.

**Definition:**

```c
METHOD ClassifyImageColor : HRESULT
VAR_INPUT
    ipSrcImage     : ITcVnImage;
    pipDestImage   : Pointer To ITcVnImage;
    fVariance      : REAL;
    fLuminanceWeight : REAL;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image.</td>
</tr>
<tr>
<td>pipDestImage</td>
<td>Pointer To ITcVnImage</td>
<td>Returns the color similarity.</td>
</tr>
<tr>
<td>fVariance</td>
<td>REAL</td>
<td>Allowed color variance.</td>
</tr>
<tr>
<td>fLuminanceWeight</td>
<td>REAL</td>
<td>Weight the impact of the luminance.</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### 6.3.3.2 ITcVnMlModel

Interface for a machine learning model.
Inheritance Hierarchy

ITcUnknown [399]
ITcVnMlModel

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>GetTypeGuid [393]</td>
<td>ITcVnMlModel</td>
<td>Gets the GUID of the model.</td>
</tr>
<tr>
<td>GetTypeName [394]</td>
<td>ITcVnMlModel</td>
<td>Gets the model type name as a string.</td>
</tr>
</tbody>
</table>

Further information

The interface ITcVnMlModel is for handling Machine Learning models. At present this only includes the use of the interface ITcVnColorModel [391].

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.3.3.2.1 GetTypeGuid

 Gets the GUID of the model.

Syntax

Definition:

METHOD GetTypeGuid : HRESULT
VAR_INPUT
  nTypeGuid : Reference To GUID;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nTypeGuid</td>
<td>Reference To GUID</td>
<td>Returns the GUID of the model.</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]
6.3.3.2.2  GetTypeName

**Definition:**

METHOD GetTypeName : HRESULT
VAR_INPUT
   sTypeName : STRING;
   nMaxLen   : UINT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sTypeName</td>
<td>STRING</td>
<td>Returns the model type name as a string.</td>
</tr>
<tr>
<td>nMaxLen</td>
<td>UINT</td>
<td>Maximum string length allowed to be written in sTypeName.</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

6.3.4  Miscellaneous

This group contains other interfaces.

6.3.4.1  ITcVnData

Interface for accessing data.

**Inheritance Hierarchy**

ITcUnknown [399]

ITcVnData
Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef</td>
<td>ITCUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface</td>
<td>ITCUnknown</td>
<td>Get a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease</td>
<td>ITCUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>CheckIfCopy</td>
<td>ITCvData</td>
<td>Check if the data are a copy (returns S_OK for copies and S_FALSE otherwise).</td>
</tr>
<tr>
<td>CheckIfImage</td>
<td>ITCvData</td>
<td>Check if the data belongs to an image (returns S_OK for images and S_FALSE otherwise).</td>
</tr>
<tr>
<td>GetSize</td>
<td>ITCvData</td>
<td>Gets the size of the data.</td>
</tr>
<tr>
<td>CheckDataPointer</td>
<td>ITCvData</td>
<td>Check if the data pointer is different from 0 (returns S_OK for non-zero data pointers and S_FALSE otherwise).</td>
</tr>
<tr>
<td>GetDataPointer</td>
<td>ITCvData</td>
<td>Gets the data pointer (the obtained data pointer must be released by calling ReleaseDataPointer).</td>
</tr>
<tr>
<td>ReleaseDataPointer</td>
<td>ITCvData</td>
<td>Release the data pointer.</td>
</tr>
</tbody>
</table>

Further information

The interface ITCvData is used for internal purposes and is not normally needed by the user. For access to raw image data, use the function GetRowPointer of the interface ITCvImage instead.

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.3.4.1.1 CheckIfCopy

Check if the data are a copy (returns S_OK for copies and S_FALSE otherwise).

Syntax

Definition:

METHOD CheckIfCopy : HRESULT

Return value

HRESULT [135]

6.3.4.1.2 CheckIfImage

Check if the data belongs to an image (returns S_OK for images and S_FALSE otherwise).
### GetSize

**Syntax**

**Definition:**

METHOD GetSize : HRESULT

**Input**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nSize</td>
<td>Reference To ULINT</td>
<td>Size of the stored data in bytes.</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

### CheckDataPointer

**Syntax**

**Definition:**

METHOD CheckDataPointer : HRESULT

**Return value**

HRESULT [135]
6.3.4.1.5 GetDataPointer

```
GetDataPointer

ppData Pointer To PVOID HRESULT GetDataPointer
```

Gets the data pointer (the obtained data pointer must be released by calling ReleaseDataPointer).

**Syntax**

**Definition:**

```
METHOD GetDataPointer : HRESULT
VAR_INPUT
  ppData : Pointer To PVOID;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ppData</td>
<td>Pointer To PVOID</td>
<td>Returns the data pointer.</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

6.3.4.1.6 ReleaseDataPointer

```
ReleaseDataPointer

ppData Pointer To PVOID HRESULT ReleaseDataPointer
```

Release the data pointer.

**Syntax**

**Definition:**

```
METHOD ReleaseDataPointer : HRESULT
VAR_INPUT
  ppData : Pointer To PVOID;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ppData</td>
<td>Pointer To PVOID</td>
<td>Pointer to the data pointer to be released.</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

6.3.4.2 ITcVnTimestamp

Offers an interface for timestamps.
Inheritance Hierarchy

ITcUnknown [399]

ITcVnTimestamp

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>ITcUnknown</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>ITcUnknown</td>
<td>Gets a reference to an implemented interface.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>ITcUnknown</td>
<td>Decrements the reference counter.</td>
</tr>
<tr>
<td>UpdateTimestamp [398]</td>
<td>ITcVnTimestamp</td>
<td>Updates the timestamp to the current time</td>
</tr>
<tr>
<td>GetTimestamp [398]</td>
<td>ITcVnTimestamp</td>
<td>Gets the timestamp</td>
</tr>
</tbody>
</table>

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.3.4.2.1 UpdateTimestamp

Updates the timestamp to the current time

Syntax

Definition:

METHOD UpdateTimestamp : HRESULT

Return value

HRESULT [135]

6.3.4.2.2 GetTimestamp

Gets the timestamp

Syntax

Definition:

METHOD GetTimestamp : HRESULT
VAR_INPUT
  nTimestamp : Reference To LINT;
END_VAR
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nTimestamp</td>
<td>Reference To LINT</td>
<td>Returns the timestamp</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.3.4.3 Interface ITcUnknown

ITcUnknown defines the reference counting as well as querying a reference to a more specific interface.

Syntax

```
TCOM_DECL_INTERFACE("00000001-0000-0000-e000-000000000064", ITcUnknown)
```

Declared in: TcInterfaces.h

Required include: -

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcAddRef [399]</td>
<td>Increments the reference counter.</td>
</tr>
<tr>
<td>TcQueryInterface [399]</td>
<td>Query of the reference to an implemented interface via the IID.</td>
</tr>
<tr>
<td>TcRelease [400]</td>
<td>Decrements the reference counter.</td>
</tr>
</tbody>
</table>

Remarks

Every TcCOM interface is directly or indirectly derived from ITcUnknown. As a consequence every TcCOM module class implements ITcUnknown, because it is derived from ITCOM Object.

The default implementation for ITcUnknown will delete the object if its last reference is released. Therefore an interface pointer must not be dereferenced after TcRelease() has been called.

6.3.4.3.1 Method ITcUnknown:TcAddRef

This method increments the reference counter.

Syntax

```
ULONG TcAddRef( )
```

Return Value

Resulting reference count value.

Description

Increments the reference counter and returns the new value.

6.3.4.3.2 Method ITcUnknown:TcQueryInterface

Query of an interface pointer with regard to an interface that is given by interface ID (IID).
### Syntax

```c
HRESULT TcQueryInterface(RITCID iid, PPVOID pipItf )
```

**iid**: (Type: RITCID) Interface IID.

**pipItf**: (PPVOID Type) pointer to interface pointer. Is set when the requested interface type is available from the corresponding instance.

### Return value

If successful, S_OK (“0”) or another positive value will be returned, cf. Return values. Extended messages refer in particular to the column HRESULT in ADS Return Codes [1486].

If the demanded interface is not available, the method returns ADSERR_DEVICE_NOINTERFACE.

### Description

Query reference to an implemented interface by the IID. It is recommended to use smart pointers to initialize and hold interface pointers.

#### Variant 1:

```c
HRESULT GetTraceLevel(ITcUnkown* ip, TcTraceLevel& tl)
{
    HRESULT hr = S_OK;
    if (ip != NULL)
    {
        ITComObjectPtr spObj;
        hr = ip->TcQueryInterface(spObj.GetIID(), &spObj);
        if (SUCCEEDED(hr))
        {
            hr = spObj->TcGetObjPara(PID_TcTraceLevel, &tl, sizeof(tl));
        }
        return hr;
    }
}
```

The interface id associated with the smart pointer can be used as parameter in TcQueryInterface. The operator “&” will return pointer to internal interface pointer member of the smart pointer. Variant 1 assumes that interface pointer is initialized if TcQueryInterface indicates success. If scope is left the destructor of the smart pointer spObj releases the reference.

#### Variant 2:

```c
HRESULT GetTraceLevel(ITcUnkown* ip, TcTraceLevel& tl)
{
    HRESULT hr = S_OK;
    ITComObjectPtr spObj = ip;
    if (spObj != NULL)
    {
        spObj->TcGetObjParam(PID_TcTraceLevel, &tl);
    }
    else
    {
        hr = ADS_E_NOINTERFACE;
    }
    return hr;
}
```

When assigning interface pointer ip to smart pointer spObj method TcQueryInterface is implicitly called with IID_ITComObject on the instance ip refers to. This results in shorter code, however it loses the original return code of TcQueryInterface.

### Method ITcUnknown::TcRelease

This method decrements the reference counter.
Syntax
ULONG TcRelease( )

Return Value
Resulting reference count value.

Description
Decrements the reference counter and returns the new value.
If reference counter gets zero, object deletes itself.

6.4 Functions
The image processing algorithms are encapsulated in functions. More complex algorithms have been
implemented as function blocks and can be found at ImageProcessing[1353].

Structure of a function call
Functions are structured as follows:
hr := F_VN_ProcessImage(<...>, hr);

Status variable of type HRESULT
Naming convention of vision API functions
Parameters and return values as references

<table>
<thead>
<tr>
<th>hr</th>
<th>Status variable [135] of the type HRESULT. The execution status of every function can be followed with this.</th>
</tr>
</thead>
<tbody>
<tr>
<td>F_VN_</td>
<td>Functions are identified by the prefix &quot;F_VN_.&quot;</td>
</tr>
<tr>
<td>Parameter &lt;...&gt;</td>
<td>Function parameters essentially fulfill the following purpose:</td>
</tr>
<tr>
<td></td>
<td>• Input parameters (e.g. images or containers) that are to be processed.</td>
</tr>
<tr>
<td></td>
<td>• Parameters for setting the algorithm</td>
</tr>
<tr>
<td></td>
<td>• Parameters as reference for returning the results</td>
</tr>
</tbody>
</table>

Grouping of functions
All functions of the TwinCAT Vision library are sorted thematically into the following groups:

Images
• Basic Image Operations [713]
• Algebraic Image Operations [467]
• Image Segmentation [1161]
• Image Color and Contrast Processing [1078]
• Image Analysis [1026]
• Image Filtering [1111]

Container
• Basic Container Operations [510]
• Algebraic Container Operations [427]
• Container Statistics [770]
API reference

Contours
• Contour Analysis [852]

Measuring
• Measurement [1215]
• Geometric and coordinate transformations [942]

Code reading
• Code Reading [753]

Drawing and text
• Drawing [883]

Advanced Functions
• Fourier Analysis [927]
• Keypoint Features [1169]

Miscellaneous
• Control Functions [878]
• Miscellaneous [1263]

Expert functions
Expert versions exist for many functions. These contain additional parameters and are usually marked by the name suffix "Exp". In the case of functions with expert versions, this is additionally noted under the heading "Expert Parameters".

For example, the expert version of F_VN_MaxImage [492] is called F_VN_MaxImagesExp [492] and contains the additional parameter ipMask.

Runtime behavior
The execution duration of some image processing algorithms depends on their parameterization and also the image content. It is therefore not deterministic. In order to guarantee real-time behavior despite that, watchdogs [137] can be applied to these functions or function groups that abort the functions after a specified time.

⚠️ WARNING
TwinCAT crash when functions are used incorrectly
If some functions are used incorrectly, this can lead to errors (and to resulting TwinCAT crashes). These are not intercepted within the function, as corresponding checks would considerably lengthen the execution time. Therefore, you must take care to use the functions properly! For example, we recommend carrying out a validation of the input parameters.

6.4.1 License Overview

The following list shows the assignment of TwinCAT Vision functions to available licenses (see Licensing model [12]). It shows which license is required for the application of certain functions.

TF7100
The basic license TC3 Vision Base (TF7100) is always required.
TC3 Vision Base

F_VN_AdaptiveThreshold [1162]
F_VN_AdaptiveThresholdExp [1163]
F_VN_AddContainers [453]
F_VN_AddImages [468]
F_VN_AddImagesWeighted [469]
F_VN_AddImagesWeightedExp [470]
F_VN_AddScalarToImage [471]
F_VN_AddToContainerElements_DINT [428]
F_VN_AddToContainerElements_INT [429]
F_VN_AddToContainerElements_LREAL [429]
F_VN_AddToContainerElements_REAL [430]
F_VN_AddToContainerElements_SINT [431]
F_VN_AddToContainerElements_TcVnKeyPoint [432]
F_VN_AddToContainerElements_TcVnPoint2_DINT [433]
F_VN_AddToContainerElements_TcVnPoint2_LREAL [433]
F_VN_AddToContainerElements_TcVnPoint2_REAL [434]
F_VN_AddToContainerElements_TcVnPoint3_LREAL [435]
F_VN_AddToContainerElements_TcVnPoint3_REAL [436]
F_VN_AddToContainerElements_TcVnRectangle_DINT [436]
F_VN_AddToContainerElements_TcVnVector2_DINT [437]
F_VN_AddToContainerElements_TcVnVector2_INT [438]
F_VN_AddToContainerElements_TcVnVector2_REAL [439]
F_VN_AddToContainerElements_TcVnVector2_SINT [439]
F_VN_AddToContainerElements_TcVnVector2_UINT [440]
F_VN_AddToContainerElements_TcVnVector2_USINT [441]
F_VN_AddToContainerElements_TcVnVector3_INT [442]
F_VN_AddToContainerElements_TcVnVector3_REAL [442]
F_VN_AddToContainerElements_TcVnVector3_SINT [443]
F_VN_AddToContainerElements_TcVnVector3_UINT [444]
F_VN_AddToContainerElements_TcVnVector3_USINT [444]
F_VN_AddToContainerElements_TcVnVector4_DINT [445]
F_VN_AddToContainerElements_TcVnVector4_INT [446]
API reference

F_VN_AddToContainerElements_TcVnVector4_LREAL [447]
F_VN_AddToContainerElements_TcVnVector4_SINT [448]
F_VN_AddToContainerElements_TcVnVector4_UINT [448]
F_VN_AddToContainerElements_TcVnVector4_USINT [449]
F_VN_AddToContainerElements_UDINT [450]
F_VN_AddToContainerElements_UINT [451]
F_VN_AddToContainerElements_ULINT [451]
F_VN_AddToContainerElements_USINT [452]
F_VN_AddVectorToImage [472]
F_VN_AdjustActiveContour [1028]
F_VN_AdvanceIterator [685]
F_VN_AppendToContainer_DINT [511]
F_VN_AppendToContainer_INT [512]
F_VN_AppendToContainer_ITcVnContainer [512]
F_VN_AppendToContainer_ITcVnForwardIterator [513]
F_VN_AppendToContainer_LREAL [514]
F_VN_AppendToContainer_REAL [515]
F_VN_AppendToContainer_SINT [515]
F_VN_AppendToContainer_TcVnDMatch [516]
F_VN_AppendToContainer_TcVnKeyPoint [517]
F_VN_AppendToContainer_TcVnPoint2_DINT [518]
F_VN_AppendToContainer_TcVnPoint2_LREAL [518]
F_VN_AppendToContainer_TcVnPoint2_REAL [519]
F_VN_AppendToContainer_TcVnPoint3_LREAL [520]
F_VN_AppendToContainer_TcVnPoint3_REAL [521]
F_VN_AppendToContainer_TcVnRectangle_DINT [521]
F_VN_AppendToContainer_TcVnVector2_DINT [522]
F_VN_AppendToContainer_TcVnVector2_INT [523]
F_VN_AppendToContainer_TcVnVector2_REAL [523]
F_VN_AppendToContainer_TcVnVector2_SINT [524]
F_VN_AppendToContainer_TcVnVector2_UINT [525]
F_VN_AppendToContainer_TcVnVector2_USINT [526]
F_VN_AppendToContainer_TcVnVector3_INT [527]
F_VN_AppendToContainer_TcVnVector3_REAL [527]
F_VN_AppendToContainer_TcVnVector3_SINT [528]
F_VN_AppendToContainer_TcVnVector3_UINT [529]
F_VN_AppendToContainer_TcVnVector3_USINT [530]
F_VN_AppendToContainer_TcVnVector4_DINT [531]
F_VN_AppendToContainer_TcVnVector4_INT [531]
F_VN_AppendToContainer_TcVnVector4_LREAL [532]
F_VN_AppendToContainer_TcVnVector4_SINT [533]
F_VN_AppendToContainer_TcVnVector4_UINT [533]
F_VN_AppendToContainer_TcVnVector4_USINT [534]
F_VN_AppendToContainer_UDINT [535]
F_VN_AppendToContainer_UINT [536]
F_VN_AppendToContainer_ULINT [536]
F_VN_AppendToContainer_USINT [537]
F_VN_ApplyColorMap [1079]
F_VN_ApplyLut [1080]
F_VN_ApplyRotationToAffineTransformation [977]
F_VN_ApplyRotationToAffineTransformationExp [978]
F_VN_ApplyScalingToAffineTransformation [978]
F_VN_ApplyScalingToAffineTransformationExp [979]
F_VN_ApplyTranslationToAffineTransformation [980]
F_VN_ApplyTranslationToAffineTransformationExp [981]
F_VN_ApplyYAxisInversionToAffineTransformation [982]
F_VN_ApplyYAxisInversionToAffineTransformationExp [983]
F_VN_ApproximatePolygon [853]
F_VN_BilateralFilter [1112]
F_VN_BilateralFilterExp [1115]
F_VN_BitwiseAndContainers [454]
F_VN_BitwiseAndImages [473]
F_VN_BitwiseAndImagesExp [473]
F_VN_BitwiseAndScalarWithImage [474]
F_VN_BitwiseAndVectorWithImage [475]
F_VN_BitwiseNotContainer [454]
F_VN_BitwiseNotImage [476]
F_VN_BitwiseNotImageExp [477]
F_VN_BitwiseOrContainers [455]
F_VN_BitwiseOrImages [477]
F_VN_BitwiseOrImagesExp [478]
F_VN_BitwiseOrScalarWithImage [479]
F_VN_BitwiseOrVectorWithImage [480]
F_VN_BitwiseXorContainers [456]
F_VN_BitwiseXorImages [481]
F_VN_BitwiseXorImagesExp [482]
F_VN_BitwiseXorScalarWithImage [483]
F_VN_BitwiseXorVectorWithImage [483]
F_VN_BinImages [484]
F_VN_BoxFilter [1117]
F_VN_BoxFilterExp [1118]
F_VN_BrightBorderObjects [1119]
F_VN_CalibrateCameraPlanar [953]
F_VN_CalibrateCameraPlanarExp [954]
F_VN_CannyEdgeDetection [1055]
F_VN_CannyEdgeDetectionExp [1057]
F_VN_CheckColorRange [1165]
F_VN_CheckIfEmpty [686]
F_VN_CheckIfIteratorIsAtEnd [687]
F_VN_CheckIfPointIsInsideContour [854]
F_VN_Clahe [1081]
F_VN_ClaheExp [1082]
F_VN_ClipLineToBoundary [1264]
F_VN_ClipLineToBoundaryITcVnImage [1265]
F_VN_ClipLineToBoundary_TcVnRectangle_DINT [1266]
F_VN_CombineImageChannels [714]
F_VN_ConnectedComponents [1058]
F_VN_ConnectedComponentsExp [1059]
F_VN_ConnectedComponentsWithStats [1060]
<table>
<thead>
<tr>
<th>Function Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>F_VN_ConnectedComponentsWithStatsExp</td>
<td>1061</td>
</tr>
<tr>
<td>F_VN_ContrainerAverage</td>
<td>840</td>
</tr>
<tr>
<td>F_VN_ContrainerAverageElementwise2</td>
<td>841</td>
</tr>
<tr>
<td>F_VN_ContrainerAverageElementwise3</td>
<td>842</td>
</tr>
<tr>
<td>F_VN_ContrainerAverageElementwise4</td>
<td>843</td>
</tr>
<tr>
<td>F_VN_ContrainerAverageVariance</td>
<td>845</td>
</tr>
<tr>
<td>F_VN_ContrainerAverageVarianceElementwise2</td>
<td>846</td>
</tr>
<tr>
<td>F_VN_ContrainerAverageVarianceElementwise3</td>
<td>847</td>
</tr>
<tr>
<td>F_VN_ContrainerAverageVarianceElementwise4</td>
<td>848</td>
</tr>
<tr>
<td>F_VN_ContourArea</td>
<td>855</td>
</tr>
<tr>
<td>F_VN_ContourCenterOfMass</td>
<td>856</td>
</tr>
<tr>
<td>F_VN_CentorCircularity</td>
<td>856</td>
</tr>
<tr>
<td>F_VN_CentorEccentricity</td>
<td>857</td>
</tr>
<tr>
<td>F_VN_CentorExtremePoint</td>
<td>858</td>
</tr>
<tr>
<td>F_VN_CentorMoments</td>
<td>859</td>
</tr>
<tr>
<td>F_VN_CentorPerimeter</td>
<td>860</td>
</tr>
<tr>
<td>F_VN_CentorRoundness</td>
<td>861</td>
</tr>
<tr>
<td>F_VN_ConvertCartesianToPolarAngleImage</td>
<td>984</td>
</tr>
<tr>
<td>F_VN_ConvertCartesianToPolarAngleImageExp</td>
<td>984</td>
</tr>
<tr>
<td>F_VN_ConvertCartesianToPolarAngles</td>
<td>985</td>
</tr>
<tr>
<td>F_VN_ConvertCartesianToPolarAnglesExp</td>
<td>986</td>
</tr>
<tr>
<td>F_VN_ConvertCartesianToPolarImages</td>
<td>987</td>
</tr>
<tr>
<td>F_VN_ConvertCartesianToPolarImagesExp</td>
<td>988</td>
</tr>
<tr>
<td>F_VN_ConvertCartesianToPolarMagnitudeImage</td>
<td>989</td>
</tr>
<tr>
<td>F_VN_ConvertCartesianToPolarMagnitudes</td>
<td>990</td>
</tr>
<tr>
<td>F_VN_ConvertCartesianToPolarPoints</td>
<td>991</td>
</tr>
<tr>
<td>F_VN_ConvertCartesianToPolarPointsExp</td>
<td>992</td>
</tr>
<tr>
<td>F_VN_ConvertColorSpace</td>
<td>1083</td>
</tr>
<tr>
<td>F_VN_ConvertContainerType</td>
<td>688</td>
</tr>
<tr>
<td>F_VN_ConvertElementType</td>
<td>715</td>
</tr>
<tr>
<td>F_VN_ConvertElementTypeExp</td>
<td>716</td>
</tr>
<tr>
<td>F_VN_ConvertITcUnknownToITcVnBitmapExport</td>
<td>1267</td>
</tr>
<tr>
<td>F_VN_ConvertITcUnknownToITcVnContainer</td>
<td>1268</td>
</tr>
<tr>
<td>Function</td>
<td>Page</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>F_VN_ConvertITcUnknownToITcVnImage</td>
<td>1269</td>
</tr>
<tr>
<td>F_VN_ConvertITcUnknownToITcVnMlModel</td>
<td>1270</td>
</tr>
<tr>
<td>F_VN_ConvertPolarToCartesianImages</td>
<td>993</td>
</tr>
<tr>
<td>F_VN_ConvertPolarToCartesianImagesExp</td>
<td>994</td>
</tr>
<tr>
<td>F_VN_ConvertPolarToCartesianPoints</td>
<td>995</td>
</tr>
<tr>
<td>F_VN_ConvertPolarToCartesianPointsExp</td>
<td>996</td>
</tr>
<tr>
<td>F_VN_ConvexHullPoints</td>
<td>862</td>
</tr>
<tr>
<td>F_VN_ConvexHullPointsExp</td>
<td>863</td>
</tr>
<tr>
<td>F_VN_ConvexityDefects</td>
<td>864</td>
</tr>
<tr>
<td>F_VN_CopyContainer</td>
<td>689</td>
</tr>
<tr>
<td>F_VN_CopyContainerElementsConditional_ITcVnContainer</td>
<td>690</td>
</tr>
<tr>
<td>F_VN_CopyContainerElementsConditional_ITcVnForwardIterator</td>
<td>691</td>
</tr>
<tr>
<td>F_VN_CopyImage</td>
<td>717</td>
</tr>
<tr>
<td>F_VN_CopyImageRegion</td>
<td>718</td>
</tr>
<tr>
<td>F_VN_CopyImageRegionToRegion</td>
<td>720</td>
</tr>
<tr>
<td>F_VN_CopyIntoDisplayableImage</td>
<td>721</td>
</tr>
<tr>
<td>F_VN_CountNonZeroPixels</td>
<td>1062</td>
</tr>
<tr>
<td>F_VN_CreateAssociatedImage</td>
<td>722</td>
</tr>
<tr>
<td>F_VN_CreateBandpassButterworthFilter</td>
<td>928</td>
</tr>
<tr>
<td>F_VN_CreateBandpassGaussianFilter</td>
<td>929</td>
</tr>
<tr>
<td>F_VN_CreateBandrejectButterworthFilter</td>
<td>930</td>
</tr>
<tr>
<td>F_VN_CreateBandrejectGaussianFilter</td>
<td>932</td>
</tr>
<tr>
<td>F_VN_CreateContainer</td>
<td>693</td>
</tr>
<tr>
<td>F_VN_CreateContainerFromArray</td>
<td>694</td>
</tr>
<tr>
<td>F_VN_CreateEmptyImage</td>
<td>723</td>
</tr>
<tr>
<td>F_VN_CreateHighpassButterworthFilter</td>
<td>933</td>
</tr>
<tr>
<td>F_VN_CreateHighpassGaussianFilter</td>
<td>934</td>
</tr>
<tr>
<td>F_VN_CreateImage</td>
<td>724</td>
</tr>
<tr>
<td>F_VN_CreateImageAndSetPixels</td>
<td>725</td>
</tr>
<tr>
<td>F_VN_CreateImageFromArray</td>
<td>726</td>
</tr>
<tr>
<td>F_VN_CreateLowpassButterworthFilter</td>
<td>935</td>
</tr>
<tr>
<td>F_VN_CreateLowpassGaussianFilter</td>
<td>936</td>
</tr>
<tr>
<td>F_VN_CreateStructuringElement</td>
<td>1121</td>
</tr>
</tbody>
</table>
F_VN_CustomElementWiseContainerOperation(ITcVnContainer) [864]
F_VN_CustomElementWiseContainerOperation(ITcVnForwardIterator) [865]
F_VN_CustomFilter() [1122]
F_VN_CustomFilterExp() [1126]
F_VN_DarkBorderObjects() [1128]
F_VN_DecomposeHomography() [997]
F_VN_DecomposeHomographyExp() [999]
F_VN_DetectBlobs() [1029]
F_VN_DetectBlobsExp() [1032]
F_VN_DivideContainers() [457]
F_VN_DivideContainersExp() [458]
F_VN_DivideImageByScalar() [485]
F_VN_DivideImageByVector() [486]
F_VN_DivideImageByVector() [487]
F_VN_DivideScalarByImage() [488]
F_VN_DivideVectorByImage() [489]
F_VN_DrawCircle() [884]
F_VN_DrawCircleExp() [885]
F_VN_DrawCircles() [886]
F_VN_DrawCirclesExp() [887]
F_VN_DrawCircularArc() [888]
F_VN_DrawCircularArcExp() [889]
F_VN_DrawContours() [890]
F_VN_DrawContoursExp() [892]
F_VN_DrawEllipse() [894]
F_VN_DrawEllipseExp() [895]
F_VN_DrawLine() [898]
F_VN_DrawLine_TcVnVector4_DINT [899]
F_VN_DrawLine_TcVnVector4_LREAL [900]
F_VN_DrawLineExp() [901]
F_VN_DrawLineExp_TcVnVector4_DINT [902]
F_VN_DrawLineExp_TcVnVector4_LREAL [903]
F_VN_DrawLines [904]
F_VN_DrawLinesExp [905]
F_VN_DrawPoint [909]
F_VN_DrawPointExp [910]
F_VN_DrawPoints [911]
F_VN_DrawPointsExp [912]
F_VN_DrawRectangle [913]
F_VN_DrawRectangle_TcVnRectangle_UDINT [914]
F_VN_DrawRotatedRectangle [915]
F_VN_DrawRotatedRectangleExp [916]
F_VN_ElementwiseExp [490]
F_VN_ElementwiseLog [491]
F_VN_EnclosingCircle [866]
F_VN_EnclosingRectangle [867]
F_VN_EnclosingTriangle [868]
F_VN_EraseFromContainer [695]
F_VN_ExportContainer [696]
F_VN_ExportContainer_String [697]
F_VN_ExportContainerSize [698]
F_VN_ExportImage [727]
F_VN_ExportImageAsBmp [728]
F_VN_ExportImageAsBmpExp [729]
F_VN_ExportImageAsBmpSize [730]
F_VN_ExportImageAsBmpSizeExp [730]
F_VN_ExportImageSize [731]
F_VN_ExportSubContainer [699]
F_VN_ExportSubContainer_String [700]
F_VN_ExportSubContainerSize [702]
F_VN_ExtractContainerRange [703]
F_VN_FillCircle [917]
F_VN_FillContainer_DINT [538]
F_VN_FillContainer INT [539]
F_VN_FillContainer LREAL [539]
F_VN_FillContainer REAL [540]
F_VN_FillContainer SINT [541]
F_VN_FillContainer TcVnDMatch [542]
F_VN_FillContainer TcVnKeyPoint [543]
F_VN_FillContainer TcVnPoint2_DINT [543]
F_VN_FillContainer TcVnPoint2_LREAL [544]
F_VN_FillContainer TcVnPoint2_REAL [545]
F_VN_FillContainer TcVnPoint3_LREAL [546]
F_VN_FillContainer TcVnPoint3_REAL [546]
F_VN_FillContainer TcVnRectangle_DINT [547]
F_VN_FillContainer TcVnVector2_DINT [548]
F_VN_FillContainer TcVnVector2_INT [549]
F_VN_FillContainer TcVnVector2_REAL [549]
F_VN_FillContainer TcVnVector2_SINT [550]
F_VN_FillContainer TcVnVector2_UINT [551]
F_VN_FillContainer TcVnVector2_USINT [552]
F_VN_FillContainer TcVnVector3_INT [552]
F_VN_FillContainer TcVnVector3_REAL [553]
F_VN_FillContainer TcVnVector3_SINT [554]
F_VN_FillContainer TcVnVector3_UINT [555]
F_VN_FillContainer TcVnVector3_USINT [555]
F_VN_FillContainer TcVnVector4_DINT [556]
F_VN_FillContainer TcVnVector4_INT [557]
F_VN_FillContainer TcVnVector4_LREAL [558]
F_VN_FillContainer TcVnVector4_SINT [558]
F_VN_FillContainer TcVnVector4_UINT [559]
F_VN_FillContainer TcVnVector4_USINT [560]
F_VN_FillContainer UDINT [561]
F_VN_FillContainer UINT [561]
F_VN_FillContainer ULINT [562]
F_VN_FillContainer USINT [563]
F_VN_FillContainerExp_DINT [564]
F_VN_FillContainerExp_INT [565]
F_VN_FillContainerExp_LREAL [565]
F_VN_FillContainerExp_REAL [566]
F_VN_FillContainerExp_SINT [567]
F_VN_FillContainerExp_TcVnDMatch [568]
F_VN_FillContainerExp_TcVnKeyPoint [569]
F_VN_FillContainerExp_TcVnPoint2_DINT [570]
F_VN_FillContainerExp_TcVnPoint2_LREAL [571]
F_VN_FillContainerExp_TcVnPoint2_REAL [572]
F_VN_FillContainerExp_TcVnPoint3_LREAL [573]
F_VN_FillContainerExp_TcVnPoint3_REAL [574]
F_VN_FillContainerExp_TcVnRectangle_DINT [575]
F_VN_FillContainerExp_TcVnVector2_DINT [576]
F_VN_FillContainerExp_TcVnVector2_INT [577]
F_VN_FillContainerExp_TcVnVector2_REAL [578]
F_VN_FillContainerExp_TcVnVector2_SINT [579]
F_VN_FillContainerExp_TcVnVector2_UINT [580]
F_VN_FillContainerExp_TcVnVector2_USINT [581]
F_VN_FillContainerExp_TcVnVector3_INT [582]
F_VN_FillContainerExp_TcVnVector3_REAL [583]
F_VN_FillContainerExp_TcVnVector3_SINT [584]
F_VN_FillContainerExp_TcVnVector3_UINT [585]
F_VN_FillContainerExp_TcVnVector3_USINT [586]
F_VN_FillContainerExp_TcVnVector4_DINT [587]
F_VN_FillContainerExp_TcVnVector4_INT [588]
F_VN_FillContainerExp_TcVnVector4_LREAL [589]
F_VN_FillContainerExp_TcVnVector4_SINT [590]
F_VN_FillContainerExp_TcVnVector4_UINT [591]
F_VN_FillContainerExp_TcVnVector4_USINT [592]
F_VN_FillContainerExp_UDINT [593]
F_VN_FillContainerExp_UINT [594]
F_VN_FillContainerExp_ULINT [595]
F_VN_FillContainerExp_USINT [596]
F_VN_FillContours [918]
F_VN_FillEllipse [919]
F_VN_FillHoles [1129]
F_VN_FillRectangle [920]
F_VN_FillRotatedRectangle [921]
F_VN_FindContourHierarchyExp [1033]
F_VN_FindContours [1037]
F_VN_FindContoursExp [1040]
F_VN_FitEllipse [869]
F_VN_FitLine [870]
F_VN_FitLineExp [870]
F_VN_FlipImage [1001]
F_VN_FourierDescriptors [871]
F_VN_FuseImages [732]
F_VN_FuseImagesArray [734]
F_VN_GaussianFilter [1131]
F_VN_GaussianFilterExp [1134]
F_VN_GenerateAffineTransformationUnitMatrix2D [1001]
F_VN_GenerateColorMap [1084]
F_VN_GenerateCustomColorMap [1086]
F_VN_GetAffineTransformation [1002]
F_VN_GetAffineTransformation2D [1003]
F_VN_GetAffineTransformation2DExp [1004]
F_VN_GetAt_DINT [597]
F_VN_GetAt_INT [598]
F_VN_GetAt_ITcVnContainer [599]
F_VN_GetAt_LREAL [600]
F_VN_GetAt_REAL [601]
F_VN_GetAt_SINT [601]
F_VN_GetAt_TcVnDMatch [602]
F_VN_GetAt_TcVnKeyPoint [603]
F_VN_GetAt_TcVnPoint2_DINT [604]
F_VN_GetAt_TcVnPoint2_LREAL [605]
F_VN_GetAt_TcVnPoint2_REAL [606]
F_VN_GetAt_TcVnPoint3_LREAL [606]
F_VN_GetAt_TcVnPoint3_REAL [607]
F_VN_GetAt_TcVnPoint2_DINT [608]
F_VN_GetAt_TcVnPoint2_INT [609]
F_VN_GetAt_TcVnPoint2_REAL [610]
F_VN_GetAt_TcVnPoint2_SINT [611]
F_VN_GetAt_TcVnPoint2_UINT [612]
F_VN_GetAt_TcVnPoint2_USINT [613]
F_VN_GetAt_TcVnPoint3_INT [614]
F_VN_GetAt_TcVnPoint3_REAL [615]
F_VN_GetAt_TcVnPoint3_SINT [616]
F_VN_GetAt_TcVnPoint3_UINT [616]
F_VN_GetAt_TcVnPoint3_USINT [617]
F_VN_GetAt_TcVnPoint4_DINT [618]
F_VN_GetAt_TcVnPoint4_INT [619]
F_VN_GetAt_TcVnPoint4_LREAL [620]
F_VN_GetAt_TcVnPoint4_SINT [621]
F_VN_GetAt_TcVnPoint4_UINT [621]
F_VN_GetAt_TcVnPoint4_USINT [622]
F_VN_GetAt_UDINT [623]
F_VN_GetAt_UINT [624]
F_VN_GetAt_ULINT [625]
F_VN_GetAt_USINT [626]
F_VN_GetContainer [703]
F_VN_GetContainerExp [704]
F_VN_GetForwardIterator [706]
F_VN_GetImageChannel [706]
F_VN_GetImageHeight [735]
F_VN_GetImageInfo [736]
F_VN_GetImageWidth [737]
F_VN_GetNumberOfElements [706]
F_VN_GetPerspectiveTransformation [1005]
F_VN_GetPixel [739]
F_VN_GetPixelFormat [739]
F_VN_GetRandomAccessIterator [707]
F_VN_GetRoi [740]
F_VN_GetTimestamp [1271]
F_VN_Histogram [1087]
F_VN_HistogramEqualization [1088]
F_VN_HistogramEqualizationExp [1089]
F_VN_HistogramExp [1090]
F_VN_Homography [1006]
F_VN_HomographyExp [1007]
F_VN_HoughCircles [1042]
F_VN_HoughCirclesExp [1043]
F_VN_HoughLines [1045]
F_VN_HoughLinesExp [1046]
F_VN_HoughLinesP [1047]
F_VN_HoughLinesPExp [1048]
F_VN_HuMomentInvariants [1271]
F_VN_ImageAverage [1066]
F_VN_ImageAverageExp [1067]
F_VN_ImageAverageStdDev [1068]
F_VN_ImageAverageStdDevExp [1069]
F_VN_ImageCenterOfMass [1070]
F_VN_ImageCenterOfMassExp [1071]
F_VN_ImageMedian [1072]
F_VN_ImageMedianExp [1073]
F_VN_ImageMoments [1074]
F_VN_IncrementIterator [708]
F_VN_InitMatrixStruct [1272]
F_VN_InsertIntoContainer_DINT [627]
F_VN_InsertIntoContainer_INT [627]
F_VN_InsertIntoContainer_ITcVnContainer [628]
F_VN_InsertIntoContainer_ITcVnForwardIterator [629]
F_VN_InsertIntoContainer_LREAL [630]
F_VN_InsertIntoContainer_REAL [631]
F_VN_InsertIntoContainer_SINT [632]
F_VN_InsertIntoContainer_TcVnDMatch [632]
F_VN_InsertIntoContainer_TcVnKeyPoint [633]
F_VN_InsertIntoContainer_TcVnPoint2_DINT [634]
F_VN_InsertIntoContainer_TcVnPoint2_LREAL [635]
F_VN_InsertIntoContainer_TcVnPoint2_REAL [636]
F_VN_InsertIntoContainer_TcVnPoint3_LREAL [636]
F_VN_InsertIntoContainer_TcVnPoint3_REAL [637]
F_VN_InsertIntoContainer_TcVnRectangle_DINT [638]
F_VN_InsertIntoContainer_TcVnVector2_DINT [639]
F_VN_InsertIntoContainer_TcVnVector2_INT [640]
F_VN_InsertIntoContainer_TcVnVector2_REAL [640]
F_VN_InsertIntoContainer_TcVnVector2_SINT [641]
F_VN_InsertIntoContainer_TcVnVector2_UINT [642]
F_VN_InsertIntoContainer_TcVnVector2_USINT [643]
F_VN_InsertIntoContainer_TcVnVector3_INT [644]
F_VN_InsertIntoContainer_TcVnVector3_REAL [644]
F_VN_InsertIntoContainer_TcVnVector3_SINT [645]
F_VN_InsertIntoContainer_TcVnVector3_UINT [646]
F_VN_InsertIntoContainer_TcVnVector3_USINT [647]
F_VN_InsertIntoContainer_TcVnVector4_DINT [648]
F_VN_InsertIntoContainer_TcVnVector4_INT [648]
F_VN_InsertIntoContainer_TcVnVector4_LREAL [649]
F_VN_InsertIntoContainer_TcVnVector4_REAL [650]
F_VN_InsertIntoContainer_TcVnVector4_SINT [651]
F_VN_InsertIntoContainer_TcVnVector4_UINT [652]
F_VN_InsertIntoContainer_UDINT [653]
F_VN_InsertIntoContainer_UINT [653]
F_VN_InsertIntoContainer_ULINT [654]
F_VN_InsertIntoContainer_USINT [655]
F_VN_InverseDft [938]
F_VN_InvertAffineTransform [1008]
F_VN_InvertColor [1092]
F_VN_InvertColorExp [1093]
F_VN_InvertMatrix3x3 [1274]
F_VN_IteratorDistance [709]
F_VN_LaplacianFilter [1135]
F_VN_LaplacianFilterExp [1138]
F_VN_LineIntersectionPoint [1275]
F_VN_LineIntersectionPointAndAngle [1275]
F_VN_LocalMaxima [1140]
F_VN_LocalMinima [1141]
F_VN_MaxContainer [459]
F_VN_MaxElement [849]
F_VN_MaxElementElementwise_DINT [770]
F_VN_MaxElementElementwise_INT [771]
F_VN_MaxElementElementwise_LREAL [772]
F_VN_MaxElementElementwise_REAL [773]
F_VN_MaxElementElementwise_SINT [773]
F_VN_MaxElementElementwise_TcVnPoint2_DINT [774]
F_VN_MaxElementElementwise_TcVnPoint2_LREAL [775]
F_VN_MaxElementElementwise_TcVnPoint2_REAL [776]
F_VN_MaxElementElementwise_TcVnPoint3_LREAL [776]
F_VN_MaxElementElementwise_TcVnVector2_DINT [777]
F_VN_MaxElementElementwise_TcVnVector2_INT [778]
F_VN_MaxElementElementwise_TcVnVector2_REAL [779]
F_VN_MaxElementElementwise_TcVnVector2_UINT [780]
F_VN_MaxElementElementwise_TcVnVector2_USINT [781]
F_VN_MaxElementElementwise_TcVnVector3_INT [782]
F_VN_MaxElementElementwise_TcVnVector3_REAL [783]
F_VN_MaxElementElementwise_TcVnVector3_SINT [784]
F_VN_MaxElementElementwise_TcVnVector3_UINT [785]
F_VN_MaxElementElementwise_TcVnVector3_USINT [785]
F_VN_MaxElementElementwise_TcVnVector4_DINT [786]
F_VN_MaxElementElementwise_TcVnVector4_INT [787]
F_VN_MaxElementElementwise_TcVnVector4_LREAL [788]
F_VN_MaxElementElementwise_TcVnVector4_SINT [788]
F_VN_MaxElementElementwise_TcVnVector4_UINT [789]
F_VN_MaxElementElementwise_TcVnVector4_USINT [790]
F_VN_MaxElementElementwise_UDINT [791]
F_VN_MaxElementElementwise_UINT [791]
F_VN_MaxElementElementwise_ULINT [792]
F_VN_MaxElementElementwise_USINT [793]
F_VN_MaxImage [492]
F_VN_MaxImageExp [492]
F_VN_MaxImageWithScalar [493]
F_VN_MaxImageWithScalarExp [494]
F_VN_MaxImageWithVector [495]
F_VN_MaxImageWithVectorExp [496]
F_VN_MaxPixelValue [1075]
F_VN_MaxPixelValueExp [1075]
F_VN_MedianElement [850]
F_VN_MedianElementElementwise_DINT [794]
F_VN_MedianElementElementwise_INT [794]
F_VN_MedianElementElementwise_LREAL [795]
F_VN_MedianElementElementwise_REAL [796]
F_VN_MedianElementElementwise_SINT [797]
F_VN_MedianElementElementwise_TcVnPoint2_DINT [797]
F_VN_MedianElementElementwise_TcVnPoint2_LREAL [798]
F_VN_MedianElementElementwise_TcVnPoint2_REAL [798]
F_VN_MedianElementElementwise_TcVnPoint3_LREAL [799]
F_VN_MedianElementElementwise_TcVnPoint3_REAL [800]
F_VN_MedianElementElementwise_TcVnVector2_DINT [801]
F_VN_MedianElementElementwise_TcVnVector2_INT [802]
F_VN_MedianElementElementwise_TcVnVector2_REAL [802]
F_VN_MedianElementElementwise_TcVnVector2_SINT [803]
F_VN_MedianElementElementwise_TcVnVector2_UINT [804]
F_VN_MedianElementElementwise_TcVnVector2_USINT [805]
F_VN_MedianElementElementwise_TcVnVector3_INT [806]
F_VN_MedianElementElementwise_TcVnVector3_REAL [806]
F_VN_MedianElementElementwise_TcVnVector3_SINT [807]
F_VN_MedianElementElementwise_TcVnVector3_UINT [808]
F_VN_MedianElementElementwise_TcVnVector3_USINT [808]
F_VN_MedianElementElementwise_TcVnVector4_DINT [809]
F_VN_MedianElementElementwise_TcVnVector4_INT [810]
F_VN_MedianElementElementwise_TcVnVector4_LREAL [811]
F_VN_MedianElementElementwise_TcVnVector4_REAL [811]
F_VN_MedianElementElementwise_TcVnVector4_SINT [812]
F_VN_MedianElementElementwise_TcVnVector4_UINT [812]
F_VN_MedianElementElementwise_TcVnVector4_USINT [813]
F_VN_MedianElementElementwise_UDINT [814]
F_VN_MedianElementElementwise_UINT [815]
F_VN_MedianElementElementwise_ULINT [815]
F_VN_MedianElementElementwise_USINT [816]
F_VN_MedianFilter [1142]
F_VN_MinContainer [460]
F_VN_MinElement [851]
F_VN_MinElementElementwise_DINT [817]
F_VN_MinElementElementwise_INT [818]
F_VN_MinElementElementwise_LREAL [818]
F_VN_MinElementElementwise_REAL [819]
F_VN_MinElementElementwise_SINT [820]
F_VN_MinElementElementwise_TcVnPoint2_DINT [821]
F_VN_MinElementElementwise_TcVnPoint2_LREAL [821]
F_VN_MinElementElementwise_TcVnPoint2_REAL [822]
F_VN_MinElementElementwise_TcVnPoint3_LREAL [823]
F_VN_MinElementElementwise_TcVnPoint3_REAL [824]
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>F_VN_MinElementElementwise_TcVnVector2_DINT</td>
<td>824</td>
</tr>
<tr>
<td>F_VN_MinElementElementwise_TcVnVector2_INT</td>
<td>825</td>
</tr>
<tr>
<td>F_VN_MinElementElementwise_TcVnVector2_REAL</td>
<td>826</td>
</tr>
<tr>
<td>F_VN_MinElementElementwise_TcVnVector2_SINT</td>
<td>827</td>
</tr>
<tr>
<td>F_VN_MinElementElementwise_TcVnVector2_UINT</td>
<td>827</td>
</tr>
<tr>
<td>F_VN_MinElementElementwise_TcVnVector2_USINT</td>
<td>828</td>
</tr>
<tr>
<td>F_VN_MinElementElementwise_TcVnVector3_INT</td>
<td>830</td>
</tr>
<tr>
<td>F_VN_MinElementElementwise_TcVnVector3_REAL</td>
<td>830</td>
</tr>
<tr>
<td>F_VN_MinElementElementwise_TcVnVector3_SINT</td>
<td>830</td>
</tr>
<tr>
<td>F_VN_MinElementElementwise_TcVnVector3_UINT</td>
<td>831</td>
</tr>
<tr>
<td>F_VN_MinElementElementwise_TcVnVector3_USINT</td>
<td>832</td>
</tr>
<tr>
<td>F_VN_MinElementElementwise_TcVnVector4_DINT</td>
<td>833</td>
</tr>
<tr>
<td>F_VN_MinElementElementwise_TcVnVector4_INT</td>
<td>833</td>
</tr>
<tr>
<td>F_VN_MinElementElementwise_TcVnVector4_LREAL</td>
<td>834</td>
</tr>
<tr>
<td>F_VN_MinElementElementwise_TcVnVector4_SINT</td>
<td>835</td>
</tr>
<tr>
<td>F_VN_MinElementElementwise_TcVnVector4_UINT</td>
<td>836</td>
</tr>
<tr>
<td>F_VN_MinElementElementwise_TcVnVector4_USINT</td>
<td>836</td>
</tr>
<tr>
<td>F_VN_MinElementElementwise_UDINT</td>
<td>837</td>
</tr>
<tr>
<td>F_VN_MinElementElementwise_UINT</td>
<td>838</td>
</tr>
<tr>
<td>F_VN_MinElementElementwise_ULINT</td>
<td>839</td>
</tr>
<tr>
<td>F_VN_MinElementElementwise_USINT</td>
<td>839</td>
</tr>
<tr>
<td>F_VN_MinImage</td>
<td>497</td>
</tr>
<tr>
<td>F_VN_MinImageExp</td>
<td>498</td>
</tr>
<tr>
<td>F_VN_MinImageWithScalar</td>
<td>499</td>
</tr>
<tr>
<td>F_VN_MinImageWithScalarExp</td>
<td>500</td>
</tr>
<tr>
<td>F_VN_MinImageWithVector</td>
<td>501</td>
</tr>
<tr>
<td>F_VN_MinImageWithVectorExp</td>
<td>502</td>
</tr>
<tr>
<td>F_VN_MinPixelValue</td>
<td>1076</td>
</tr>
<tr>
<td>F_VN_MinPixelValueExp</td>
<td>1077</td>
</tr>
<tr>
<td>F_VN_MixImageChannels</td>
<td>741</td>
</tr>
<tr>
<td>F_VN_MorphologicalOperator</td>
<td>1145</td>
</tr>
<tr>
<td>F_VN_MultiplyContainers</td>
<td>461</td>
</tr>
<tr>
<td>F_VN_MultiplyImages</td>
<td>503</td>
</tr>
</tbody>
</table>
F_VN_MultiplyImageWithScalar [503]
F_VN_MultiplyImageWithVector [504]
F_VN_MultiplyMatrices [1276]
F_VN_MultiplyWithContainerElements1 [462]
F_VN_MultiplyWithContainerElements2 [463]
F_VN_MultiplyWithContainerElements3 [464]
F_VN_MultiplyWithContainerElements4 [464]
F_VN_NegateContainer [465]
F_VN_NormalizeImage [1094]
F_VN_NormalizeImageExp [1095]
F_VN_NormalizeImageForDisplay [1096]
F_VN_OptimalDftSize [939]
F_VN_PadImageBorder [940]
F_VN_PadImageBorderExp [941]
F_VN_PerspectiveTransformation [1009]
F_VN_PutLabel [922]
F_VN_PutLabelExp [923]
F_VN_PutText [924]
F_VN_PutTextExp [926]
F_VN_PyramidDown [1010]
F_VN_PyramidUp [1011]
F_VN_ReferenceColorSimilarity_ITcVnColorModel [1097]
F_VN_ReferenceColorSimilarity_ITcVnMlModel [1098]
F_VN_ReferenceColorSimilarity_TcVnVector3_LREAL [1099]
F_VN_ReferenceColorSimilarityExp_ITcVnColorModel [1099]
F_VN_ReferenceColorSimilarityExp_ITcVnMlModel [1100]
F_VN_ReferenceColorSimilarityExp_TcVnVector3_LREAL [1101]
F_VN_RemapImageToLogPolarSpace [1012]
F_VN_RemapImageToLogPolarSpaceExp [1013]
F_VN_RemapImageToPolarSpace [1014]
F_VN_RemapImageToPolarSpaceExp [1014]
F_VN_RemoveLocalMaxima [1146]
F_VN_RemoveLocalMinima [1148]
F_VN_ResizeImage
F_VN_ResetRoi
F_VN_ReserveContainerMemory
F_VN_ReversedContainer
F_VN_RotatedRectangleCorners
F_VN_RotatedRectangleIntersection
F_VN_RotateImage
F_VN_ScharrFilter
F_VN_ScharrFilterExp
F_VN_SeparableCustomFilter
F_VN_SeparableCustomFilterExp
F_VN_SetAt_DINT
F_VN_SetAt_INT
F_VN_SetAt_ITcVnContainer
F_VN_SetAt_LREAL
F_VN_SetAt_REAL
F_VN_SetAt_SINT
F_VN_SetAt_TcVnDMatch
F_VN_SetAt_TcVnKeyPoint
F_VN_SetAt_TcVnPoint2_DINT
F_VN_SetAt_TcVnPoint2_LREAL
F_VN_SetAt_TcVnPoint2_REAL
F_VN_SetAt_TcVnPoint3_LREAL
F_VN_SetAt_TcVnPoint3_REAL
F_VN_SetAt_TcVnRectangle_DINT
F_VN_SetAt_TcVnVector2_DINT
F_VN_SetAt_TcVnVector2_INT
F_VN_SetAt_TcVnVector2_REAL
F_VN_SetAt_TcVnVector2_SINT
F_VN_SetAt_TcVnVector2_UINT
F_VN_SetAt_TcVnVector2_USINT
F_VN_SetAt_TcVnVector3_INT
F_VN_SetAt_TcVnVector3_REAL
F_VN_SetAt_TcVnVector3_SINT  [674]
F_VN_SetAt_TcVnVector3_UINT  [675]
F_VN_SetAt_TcVnVector3_USINT [676]
F_VN_SetAt_TcVnVector4_DINT [677]
F_VN_SetAt_TcVnVector4_INT  [678]
F_VN_SetAt_TcVnVector4_LREAL [679]
F_VN_SetAt_TcVnVector4_SINT [680]
F_VN_SetAt_TcVnVector4_UINT [681]
F_VN_SetAt_TcVnVector4_USINT [682]
F_VN_SetAt_UDINT   [683]
F_VN_SetAt_UINT    [684]
F_VN_SetAt_ULINT   [685]
F_VN_SetAt_USINT   [686]
F_VN_SetContainer  [712]
F_VN_SetImageChannel  [743]
F_VN_SetIteratorToBegin [712]
F_VN_SetPixel      [744]
F_VN_SetPixels     [745]
F_VN_SetPixelsExp  [746]
F_VN_SetRngSeed    [1279]
F_VN_SetRoi        [747]
F_VN_SetRoi_TcVnRectangle_UDINT [748]
F_VN_SobelFilter   [1157]
F_VN_SobelFilterExp [1159]
F_VN_SplitImageChannels [749]
F_VN_StartAbsWatchdog [878]
F_VN_StartAbsWatchdogExp [879]
F_VN_StartRelWatchdog [880]
F_VN_StartRelWatchdogExp [881]
F_VN_StopWatchdog   [882]
F_VN_SubtractContainers [466]
F_VN_SubtractImageFromScalar [505]
F_VN_SubtractImageFromVector [506]
F_VN_SubtractImages [507]
F_VN_SubtractScalarFromImage [508]
F_VN_SubtractVectorFromImage [509]
F_VN_Threshold [1166]
F_VN_TrainImageColor [1102]
F_VN_TrainImageColor_ITcVnMIModel [1104]
F_VN_TrainImageColorExp [1105]
F_VN_TrainImageColorExp_ITcVnMIModel [1107]
F_VN_TrainImageColorExp2 [1108]
F_VN_TrainImageColorExp2_ITcVnMIModel [1110]
F_VN_TransformCoordinatesPlanar_Container [973]
F_VN_TransformCoordinatesPlanar_Point [974]
F_VN_TransformIntoDisplayableImage [750]
F_VN_TransformIntoDisplayableImageExp [751]
F_VN_UpdateTimestamp [1280]
F_VN_UprightBoundingRectangle [877]
F_VN_WarpAffine [1017]
F_VN_WarpAffine_Container [1018]
F_VN_WarpAffine_Point [1019]
F_VN_WarpAffineExp [1020]
F_VN_WarpPerspective [1021]
F_VN_WarpPerspective_Container [1022]
F_VN_WarpPerspective_Point [1023]
F_VN_WarpPerspective_Rectangle [1024]
F_VN_WarpPerspectiveExp [1025]
F_VN_WatershedSegmentationExp [1169]

**TC3 Vision Code Reading**

F_VN_ReadBarcode [755]
F_VN_ReadBarcodeExp [756]
F_VN_ReadDataMatrixCode [759]
F_VN_ReadDataMatrixCodeExp [761]
F_VN_ReadPharmaCode [762]
F_VN_ReadPharmaCodeExp [764]
F_VN_ReadQRCode [766]
F_VN_ReadQRCodeExp [768]

**TC3 Vision Metrology 2D**

F_VN_AdjustSearchWindowOrientationToLinearEdge [1221]
F_VN_CalibrateCamera [944]
F_VN_CalibrateCameraExp [947]
F_VN_CalibrateCameraExp2 [950]
F_VN_CalibrateCameraManually [952]
F_VN_CalibrateLinescanCamera [956]
F_VN_CalibrateLinescanCameraExp [958]
F_VN_ClosestPointsBF [1222]
F_VN_CompensateLensDistortion [960]
F_VN_CompensateLensDistortionExp1 [961]
F_VN_CompensateLensDistortionExp2 [962]
F_VN_CompensateLensDistortionForPoints [963]
F_VN_CompensateLensDistortionForPointsExp1 [964]
F_VN_CompensateLensDistortionForPointsExp2 [965]
F_VN_DetectPatternPoints [966]
F_VN_DetectPatternPointsExp [967]
F_VN_ImagePointsToWorldDistance [969]
F_VN_LocateCircularArc [1224]
F_VN_LocateCircularArcExp [1227]
F_VN_LocateEdge [1230]
F_VN_LocateEdgeExp [1233]
F_VN_LocateEdges [1236]
F_VN_LocateEdgesExp [1239]
F_VN_LocateEllipse [1242]
F_VN_LocateEllipseExp [1244]
F_VN_MeasureAngleBetweenEdges [1248]
F_VN_MeasureAngleBetweenEdgesExp [1250]
F_VN_MeasureEdgeDistance [1253]
F_VN_MeasureEdgeDistanceExp [1255]
F_VN_MeasureMinEdgeDistance [1258]
API reference

F_VN_MeasureMinEdgeDistanceExp [1260]
F_VN_SortDetectedPatternPoints [970]
F_VN_TransformCoordinatesImageToWorld_CONTAINER [971]
F_VN_TransformCoordinatesImageToWorld_POINT [972]
F_VN_TransformCoordinatesWorldToImage_CONTAINER [975]
F_VN_TransformCoordinatesWorldToImage_POINT [976]

TC3 Vision Matching
F_VN_DrawKeypoints [896]
F_VN_DrawKeypointsExp [896]
F_VN_DrawMatches [907]
F_VN_DrawMatchesExp [908]
F_VN_FilterGoodMatches [1170]
F_VN_FindReferenceKeyPointsInImage [1171]
F_VN_FindReferenceKeyPointsInImageAKAZE [1173]
F_VN_FindReferenceKeyPointsInImageAKAZEExp [1175]
F_VN_FindReferenceKeyPointsInImageBRISK [1177]
F_VN_FindReferenceKeyPointsInImageBRISKExp [1179]
F_VN_FindReferenceKeyPointsInImageExp [1181]
F_VN_FindReferenceKeyPointsInImageORB [1183]
F_VN_FindReferenceKeyPointsInImageORBExp [1185]
F_VN_GetMatchCoordinates [1187]
F_VN_KeyPointsAGAST [1188]
F_VN_KeyPointsAGASTExp [1189]
F_VN_KeyPointsAndDescriptorsAKAZE [1190]
F_VN_KeyPointsAndDescriptorsAKAZEExp [1190]
F_VN_KeyPointsAndDescriptorsBRISK [1191]
F_VN_KeyPointsAndDescriptorsBRISKExp [1192]
F_VN_KeyPointsAndDescriptorsKAZE [1193]
F_VN_KeyPointsAndDescriptorsKAZEExp [1194]
F_VN_KeyPointsAndDescriptorsORB [1195]
F_VN_KeyPointsAndDescriptorsORBExp [1196]
F_VN_KeyPointsFAST [1197]
F_VN_KeyPointsFASTExp [1198]
F_VN_KeyPointsGFTT [1199]
F_VN_KeyPointsGFTTExp [1200]
F_VN_KeyPointsMSER [1201]
F_VN_KeyPointsMSERExp [1202]
F_VN_KeyPointsSB [1203]
F_VN_KeyPointsSBExp [1204]
F_VN_MatchContours [872]
F_VN_MatchContours1vsN [873]
F_VN_MatchContours1vsNExp [875]
F_VN_MatchContoursExp [876]
F_VN_MatchDescriptorsBF [1205]
F_VN_MatchDescriptorsBFExp [1206]
F_VN_MatchDescriptorsFlannLsh [1207]
F_VN_MatchDescriptorsFlannLshExp [1208]
F_VN_MatchDescriptorsKnnBF [1209]
F_VN_MatchDescriptorsKnnBFExp [1210]
F_VN_MatchDescriptorsKnnFlannLsh [1211]
F_VN_MatchDescriptorsKnnFlannLshExp [1212]
F_VN_MatchImageHuMoments [1049]
F_VN_MatchTemplate [1050]
F_VN_MatchTemplateAndEvaluate [1051]
F_VN_MatchTemplateAndEvaluateExp [1052]
F_VN_MatchTemplateExp [1053]
F_VN_RegionsMSER [1213]
F_VN_RegionsMSERExp [1214]

### 6.4.2 Algebraic Container Operations

This group contains functions for the element-wise application of algebraic operators to containers.

#### Functions

**Algebra**

- F_VN_AddContainers [453]
- F_VN_AddToContainerElements [428]
- F_VN_DivideContainers(Exp) [457]
- F_VN_MultiplyContainers [461]
API reference

- F_VN_MultiplyWithContainerElements [› 462]
- F_VN_NegateContainer [› 465]
- F_VN_SubtractContainers [› 466]

Bitwise operators
- F_VN_BitwiseAndContainers [› 454]
- F_VN_BitwiseNotContainer [› 454]
- F_VN_BitwiseOrContainers [› 455]
- F_VN_BitwiseXorContainers [› 456]

Statistical features
- F_VN_MaxContainer [› 459]
- F_VN_MinContainer [› 460]

6.4.2.1 F_VN_AddToContainerElements
This chapter contains functions to add a value to each container element according to the data type.

6.4.2.1.1 F_VN_AddToContainerElements_DINT

Add a value to each container element.

Syntax

```
FUNCTION F_VN_AddToContainerElements_DINT : HRESULT
VAR_INPUT
  nValue : DINT;
  ipContainer : ITCvNContainer;
  hrPrev : HRESULT;
END_VAR
```

```
F_VN_AddToContainerElements_DINT

nValue DINT
ipContainer ITCvNContainer
hrPrev HRESULT
```

Return value

HRESULT [› 135]

Required License

TC3 Vision Base
### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.2.1.2 F_VN_AddToContainerElements_INT

**F_VN_AddToContainerElements_INT**

```
FUNCTION F_VN_AddToContainerElements_INT : HRESULT
VAR_INPUT
  nValue : INT;
  ipContainer : ITcVnContainer;
  hrPrev : HRESULT;
END_VAR
```

Add a value to each container element.

#### Syntax

**Definition:**

```
FUNCTION F_VN_AddToContainerElements_INT : HRESULT
VAR_INPUT
  nValue : INT;
  ipContainer : ITcVnContainer;
  hrPrev : HRESULT;
END_VAR
```

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nValue</td>
<td>INT</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with INT elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

#### Return value

**HRESULT [135]**

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.2.1.3 F_VN_AddToContainerElements_LREAL

**F_VN_AddToContainerElements_LREAL**

```
FUNCTION F_VN_AddToContainerElements_LREAL : HRESULT
VAR_INPUT
  fValue : LREAL;
  ipContainer : ITcVnContainer;
  hrPrev : HRESULT;
END_VAR
```

Add a value to each container element.
Syntax

Definition:

FUNCTION F_VN_AddToContainerElements_REAL : HRESULT
VAR_INPUT
    fValue : LREAL;
    ipContainer : ITcVnContainer;
    hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fValue</td>
<td>LREAL</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with LREAL elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.2.1.4 F_VN_AddToContainerElements_REAL

Add a value to each container element.

Syntax

Definition:

FUNCTION F_VN_AddToContainerElements_REAL : HRESULT
VAR_INPUT
    fValue : REAL;
    ipContainer : ITcVnContainer;
    hrPrev : HRESULT;
END_VAR
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nValue</td>
<td>SINT</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with SINT elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.2.1.5 F_VN_AddToContainerElements_SINT

Add a value to each container element.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_AddToContainerElements_SINT : HRESULT
VAR_INPUT
  nValue : SINT;
  ipContainer : ITcVnContainer;
  hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nValue</td>
<td>SINT</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with SINT elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]
F_VN_AddToContainerElements_TcVnKeyPoint

Add a value to each container element.

Syntax

Definition:

FUNCTION F_VN_AddToContainerElements_TcVnKeyPoint : HRESULT
VAR_INPUT
    stValue : Reference To TcVnKeyPoint;
    ipContainer : ITCvnContainer;
    hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stValue</td>
<td>Reference To TcVnKeyPoint</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvnContainer</td>
<td>Container with TcVnKeyPoint elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.2.1.7 F_VN_AddToContainerElements_TcVnPoint2_DINT

Add a value to each container element.

Syntax

Definition:
FUNCTION F_VN_AddToContainerElements_TcVnPoint2_DINT : HRESULT
VAR_INPUT
 aValue : Reference To TcVnPoint2_DINT;
 ipContainer : ITcVnContainer;
 hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aValue</td>
<td>Reference To TcVnPoint2_DINT [151]</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with TcVnPoint2_DINT elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.2.1.8 F_VN_AddToContainerElements_TcVnPoint2_LREAL

Add a value to each container element.

Syntax

Definition:
FUNCTION F_VN_AddToContainerElements_TcVnPoint2_LREAL : HRESULT
VAR_INPUT
 aValue : Reference To TcVnPoint2_LREAL;
 ipContainer : ITcVnContainer;
 hrPrev : HRESULT;
END_VAR
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aValue</td>
<td>Reference To TcVnPoint2_REAL</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with TcVnPoint2_REAL elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### Syntax

**F_VN_AddToContainerElements_TcVnPoint2_REAL**

Add a value to each container element.

**Definition:**

```
FUNCTION F_VN_AddToContainerElements_TcVnPoint2_REAL : HRESULT
   VAR_INPUT
      aValue : Reference To TcVnPoint2_REAL;
      ipContainer : ITcVnContainer;
      hrPrev : HRESULT;
      hrPrev = HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aValue</td>
<td>Reference To TcVnPoint2_REAL</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with TcVnPoint2_REAL elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]
**API reference**

**Required License**
TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.2.1.10 F_VN_AddToContainerElements_TcVnPoint3_LREAL**

Add a value to each container element.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_AddToContainerElements_TcVnPoint3_LREAL : HRESULT
VAR_INPUT
    aValue : Reference To TcVnPoint3_LREAL;
    ipContainer : ITcVnContainer;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aValue</td>
<td>Reference To TcVnPoint3_LREAL</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with TcVnPoint3_LREAL elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

**HRESULT**

**Required License**
TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.2.1.11  

**F_VN_AddToContainerElements_TcVnPoint3_REAL**

Add a value to each container element.

### Syntax

**Definition:**

```c
FUNCTION F_VN_AddToContainerElements_TcVnPoint3_REAL : HRESULT
VAR_INPUT
  aValue : Reference To TcVnPoint3_REAL;
  ipContainer : ITcVnContainer;
  hrPrev : HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aValue</td>
<td>Reference To TcVnPoint3_REAL</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with TcVnPoint3_REAL elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

**HRESULT [135]**

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

---

6.4.2.1.12  

**F_VN_AddToContainerElements_TcVnRectangle_DINT**

Add a value to each container element.

### Syntax

**Definition:**

```c
FUNCTION F_VN_AddToContainerElements_TcVnRectangle_DINT : HRESULT
VAR_INPUT
  stValue : Reference To TcVnRectangle_DINT;
  ipContainer : ITcVnContainer;
  hrPrev : HRESULT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stValue</td>
<td>Reference To TcVnRectangle_DINT</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with TcVnRectangle_DINT elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.2.1.13 F_VN_AddToContainerElements_TcVnVector2_DINT**

Add a value to each container element.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_AddToContainerElements_TcVnVector2_DINT : HRESULT
VAR_INPUT
  aValue : Reference To TcVnVector2_DINT;
  ipContainer : ITcVnContainer;
  hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector2_DINT</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with TcVnVector2_DINT elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]
Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.2.1.14  **F_VN_AddToContainerElements_TcVnVector2_INT**

Add a value to each container element.

**Syntax**

**Definition:**

```
FUNCTION F_VN_AddToContainerElements_TcVnVector2_INT : HRESULT
VAR_INPUT
    aValue      : Reference To TcVnVector2_INT;
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector2_INT</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with TcVnVector2_INT elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
API reference

6.4.2.1.15 F_VN_AddToContainerElements_TcVnVector2_REAL

Add a value to each container element.

Syntax

Definition:
FUNCTION F_VN_AddToContainerElements_TcVnVector2_REAL : HRESULT
VAR_INPUT
aValue : Reference To TcVnVector2_REAL;
ipContainer : ITCvnContainer;
hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector2_REAL [153]</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvnContainer [345]</td>
<td>Container with TcVnVector2_REAL elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.2.1.16 F_VN_AddToContainerElements_TcVnVector2_SINT

Add a value to each container element.

Syntax

Definition:
FUNCTION F_VN_AddToContainerElements_TcVnVector2_SINT : HRESULT
VAR_INPUT
aValue : Reference To TcVnVector2_SINT;
ipContainer : ITCvnContainer;
hrPrev : HRESULT;
END_VAR
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector2_SINT</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with TcVnVector2_SINT elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.2.1.17  **F_VN_AddToContainerElements_TcVnVector2_UINT**

Add a value to each container element.

**Syntax**

**Definition:**

FUNCTION F_VN_AddToContainerElements_TcVnVector2_UINT : HRESULT

VAR_INPUT

    aValue : Reference To TcVnVector2_UINT;
    ipContainer : ITcVnContainer;
    hrPrev : HRESULT;

END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector2_UINT</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with TcVnVector2_UINT elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]
6.4.2.18    F_VN_AddToContainerElements_TcVnVector2_USINT

Add a value to each container element.

Syntax

Definition:

```c
FUNCTION F_VN_AddToContainerElements_TcVnVector2_USINT : HRESULT
VAR_INPUT
    aValue : Reference To TcVnVector2_USINT;
    ipContainer : ITcVnContainer;
    hrPrev : HRESULT;
END_VAR
```

_inputs_

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector2_USINT</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with TcVnVector2_USINT elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

_Return value_

HRESULT [135]

_REQUIRED LICENSE_

TC3 Vision Base

_System Requirements_

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

TC3 Vision Base
6.4.2.1.19  F_VN_AddToContainerElements_TcVnVector3_INT

Add a value to each container element.

Syntax

Definition:
FUNCTION F_VN_AddToContainerElements_TcVnVector3_INT : HRESULT
VAR_INPUT
  aValue : Reference To TcVnVector3_INT;
  ipContainer : ITcVnContainer;
  hrPrev : HRESULT;
END_VAR

Inpusts

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector3_INT [153]</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with TcVnVector3_INT elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

Development environment | Target platform | PLC libraries to include |
------------------------|-----------------|-------------------------|
TwinCAT V3.1.4024.17 or later | PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU | Tc3_Vision |

6.4.2.1.20  F_VN_AddToContainerElements_TcVnVector3_REAL

Add a value to each container element.

Syntax

Definition:
FUNCTION F_VN_AddToContainerElements_TcVnVector3_REAL : HRESULT
VAR_INPUT
  aValue : Reference To TcVnVector3_REAL;
  ipContainer : ITcVnContainer;
  hrPrev : HRESULT;
END_VAR
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector3_REAL [153]</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer [345]</td>
<td>Container with TcVnVector3_REAL elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.2.1.21  F_VN_AddToContainerElements_TcVnVector3_SINT

Add a value to each container element.

**Syntax**

Definition:

```
FUNCTION F_VN_AddToContainerElements_TcVnVector3_SINT : HRESULT
VAR_INPUT
  aValue : Reference To TcVnVector3_SINT;
  ipContainer : ITCvNContainer;
  hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector3_SINT [153]</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer [345]</td>
<td>Container with TcVnVector3_SINT elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]
**Required License**
TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.2.1.22 F_VN_AddToContainerElements_TcVnVector3_UINT

Add a value to each container element.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_AddToContainerElements_TcVnVector3_UINT : HRESULT
VAR_INPUT
  aValue     : Reference To TcVnVector3_UINT;
  ipContainer: ITCvNContainer;
  hrPrev     : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector3_UINT</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer</td>
<td>Container with TcVnVector3_UINT elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT

**Required License**
TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.2.1.23 F_VN_AddToContainerElements_TcVnVector3_USINT

Add a value to each container element.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_AddToContainerElements_TcVnVector3_USINT : HRESULT
VAR_INPUT
  aValue     : Reference To TcVnVector3_USINT;
  ipContainer: ITCvNContainer;
  hrPrev     : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector3_USINT</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer</td>
<td>Container with TcVnVector3_USINT elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Add a value to each container element.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_AddToContainerElements_TcVnVector3_USINT : HRESULT
VAR_INPUT
    aValue     : Reference To TcVnVector3_USINT;
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector3_USINT</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with TcVnVector3_USINT elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.2.1.24 F_VN_AddToContainerElements_TcVnVector4_DINT**

```plaintext
FUNCTION F_VN_AddToContainerElements_TcVnVector4_DINT : HRESULT
VAR_INPUT
    aValue     : Reference To TcVnVector4_DINT;
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Add a value to each container element.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_AddToContainerElements_TcVnVector4_DINT : HRESULT
VAR_INPUT
    aValue     : Reference To TcVnVector4_DINT;
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector4_DINT [153]</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with TcVnVector4_DINT elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.2.1.25  F_VN_AddToContainerElements_TcVnVector4_INT

Add a value to each container element.

Syntax

Definition:

FUNCTION F_VN_AddToContainerElements_TcVnVector4_INT : HRESULT
VAR_INPUT
  aValue : Reference To TcVnVector4_INT;
  ipContainer : ITcVnContainer;
  hrPrev : HRESULT;
END_VAR
**Required License**
TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.2.1.26 F_VN_AddToContainerElements_TcVnVector4_LREAL

Add a value to each container element.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_AddToContainerElements_TcVnVector4_LREAL : HRESULT
VAR_INPUT
    aValue : Reference To TcVnVector4_LREAL;
    ipContainer : ITcVnContainer;
    hrPrev : HRESULT;
END_VAR
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with TcVnVector4_LREAL elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]
Add a value to each container element.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_AddToContainerElements_TcVnVector4_SINT : HRESULT
VAR_INPUT
    aValue      : Reference To TcVnVector4_SINT;
    ipContainer : ITCvVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector4_SINT</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvVnContainer</td>
<td>Container with TcVnVector4_SINT elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

---

### 6.4.2.1.28 F_VN_AddToContainerElements_TcVnVector4_UINT

Add a value to each container element.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_AddToContainerElements_TcVnVector4_UINT : HRESULT
VAR_INPUT
    aValue      : Reference To TcVnVector4_UINT;
    ipContainer : ITCvVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

---
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector4_UINT</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer</td>
<td>Container with TcVnVector4_UINT elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## Syntax

**Definition:**

```plaintext
FUNCTION F_VN_AddToContainerElements_TcVnVector4_USINT : HRESULT
VAR_INPUT
    aValue : Reference To TcVnVector4_USINT;
    ipContainer : ITCvNContainer;
    hrPrev : HRESULT;
END_VAR
```

Add a value to each container element.

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector4_USINT</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer</td>
<td>Container with TcVnVector4_USINT elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]
6.4.2.1.30 F_VN_AddToContainerElements_UDINT

Add a value to each container element.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_AddToContainerElements_UDINT : HRESULT
VAR_INPUT
  nValue    : UDINT;
  ipContainer : ITcVnContainer;
  hrPrev     : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nValue</td>
<td>UDINT</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with UDINT elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.2.1.31  F_VN_AddToContainerElements_UINT

Add a value to each container element.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_AddToContainerElements_UINT : HRESULT
VAR_INPUT
    nValue : UINT;
    ipContainer : ITcVnContainer;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nValue</td>
<td>UINT</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with UINT elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.2.1.32  F_VN_AddToContainerElements_ULINT

Add a value to each container element.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_AddToContainerElements_ULINT : HRESULT
VAR_INPUT
    nValue : ULINT;
    ipContainer : ITcVnContainer;
    hrPrev : HRESULT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nValue</td>
<td>USINT</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer ][ 345]</td>
<td>Container with USINT elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT][ 135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT][ 135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.2.1.33 **F_VN_AddToContainerElements_USINT**

Add a value to each container element.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_AddToContainerElements_USINT : HRESULT
VAR_INPUT
  nValue : USINT;
  ipContainer : ITcVnContainer;
  hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nValue</td>
<td>USINT</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer ][ 345]</td>
<td>Container with USINT elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT][ 135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT][ 135]
Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.2.2 F_VN_AddContainers

Element-wise addition of two containers (same length and type).

Syntax

Definition:

FUNCTION F_VN_AddContainers : HRESULT
VAR_INPUT
    ipSrcContainer1 : ITCvNContainer;
    ipSrcContainer2 : ITCvNContainer;
    ipDestContainer : Reference To ITCvNContainer;
    hrPrev            : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcContainer1</td>
<td>ITCvNContainer</td>
<td>Source container 1</td>
</tr>
<tr>
<td>ipSrcContainer2</td>
<td>ITCvNContainer</td>
<td>Source container 2</td>
</tr>
<tr>
<td>ipDestContainer</td>
<td>Reference To ITCvNContainer</td>
<td>Returns the resulting container</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.2.3 F_VN_BitwiseAndContainers

Element-wise application of a bit-wise AND operator to two containers (same length and type, integer only).

Syntax

Definition:

FUNCTION F_VN_BitwiseAndContainers : HRESULT 
VAR_INPUT
   ipSrcContainer1 : ITcVnContainer;
   ipSrcContainer2 : ITcVnContainer;
   ipDestContainer : Reference To ITcVnContainer;
   hrPrev : HRESULT;
END_VAR

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcContainer1</td>
<td>ITcVnContainer</td>
<td>Source container 1</td>
</tr>
<tr>
<td>ipSrcContainer2</td>
<td>ITcVnContainer</td>
<td>Source container 2</td>
</tr>
<tr>
<td>ipDestContainer</td>
<td>Reference To ITcVnContainer</td>
<td>Returns the resulting container</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.2.4 F_VN_BitwiseNotContainer

Element-wise application of a bit-wise NOT operator to a container (integer only).
Syntax

Definition:

```
FUNCTION F_VN_BitwiseNotContainer : HRESULT
VAR_INPUT
    ipSrcContainer : ITcVnContainer;
    ipDestContainer : Reference To ITcVnContainer;
    hrPrev          : HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>ipDestContainer</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns the resulting container</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.2.5 F_VN_BitwiseOrContainers

```
F_VN_BitwiseOrContainers
ipSrcContainer1 ITcVnContainer
ipSrcContainer2 ITcVnContainer
ipDestContainer Reference To ITcVnContainer
hrPrev HRESULT
```

Element-wise application of a bit-wise OR operator to two containers (same length and type, integer only).

### Syntax

Definition:

```
FUNCTION F_VN_BitwiseOrContainers : HRESULT
VAR_INPUT
    ipSrcContainer1 : ITcVnContainer;
    ipSrcContainer2 : ITcVnContainer;
    ipDestContainer : Reference To ITcVnContainer;
    hrPrev          : HRESULT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcContainer1</td>
<td>ITcVnContainer [345]</td>
<td>Source container 1</td>
</tr>
<tr>
<td>ipSrcContainer2</td>
<td>ITcVnContainer [345]</td>
<td>Source container 2</td>
</tr>
<tr>
<td>ipDestContainer</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns the resulting container</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.2.6 F_VN_BitwiseXorContainers**

Element-wise application of a bit-wise XOR operator to two containers (same length and type, integer only).

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_BitwiseXorContainers : HRESULT
VAR_INPUT
    ipSrcContainer1 : ITcVnContainer;
    ipSrcContainer2 : ITcVnContainer;
    ipDestContainer : Reference To ITcVnContainer;
    hrPrev : HRESULT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcContainer1</td>
<td>ITcVnContainer [345]</td>
<td>Source container 1</td>
</tr>
<tr>
<td>ipSrcContainer2</td>
<td>ITcVnContainer [345]</td>
<td>Source container 2</td>
</tr>
<tr>
<td>ipDestContainer</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns the resulting container</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.2.7 F_VN_DivideContainers**

Element-wise division of two containers (same length and type). In case of integer division by zero, the resulting element is set to 0 and S_DIVISION_BY_ZERO is returned instead of S_OK (value can be changed in F_VN_DivideContainersExp).

**Syntax**

**Definition:**

FUNCTION F_VN_DivideContainers : HRESULT
VAR_INPUT
    ipSrcContainer1 : ITcVnContainer;
    ipSrcContainer2 : ITcVnContainer;
    ipDestContainer : Reference To ITcVnContainer;
    hrPrev : HRESULT;
END_VAR
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcContainer1</td>
<td>ITcVnContainer [345]</td>
<td>Source container 1</td>
</tr>
<tr>
<td>ipSrcContainer2</td>
<td>ITcVnContainer [345]</td>
<td>Source container 2</td>
</tr>
<tr>
<td>ipDestContainer</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns the resulting container</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.2.8  **F_VN_DivideContainersExp**

Element-wise division of two containers (same length and type). In case of integer division by zero, the resulting element is set to nIntDivideByZeroResult and S_DIVISION_BY_ZERO is returned instead of S_OK. (expert function)

### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_DivideContainersExp : HRESULT
VAR_INPUT
    ipSrcContainer1 : ITcVnContainer;
    ipSrcContainer2 : ITcVnContainer;
    ipDestContainer : Reference To ITcVnContainer;
    nIntDivideByZeroResult : LINT;
    hrPrev : HRESULT;
END_VAR
```
### 6.4.2.9 F_VN_MaxContainer

Element-wise maximum of two containers (same length and type).

#### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_MaxContainer : HRESULT
VAR_INPUT
    ipSrcContainer1 : ITcVnContainer;
    ipSrcContainer2 : ITcVnContainer;
    ipDestContainer : Reference To ITcVnContainer;
    hrPrev : HRESULT;
END_VAR
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcContainer1</td>
<td>ITcVnContainer [1 345]</td>
<td>Source container 1</td>
</tr>
<tr>
<td>ipSrcContainer2</td>
<td>ITcVnContainer [1 345]</td>
<td>Source container 2</td>
</tr>
<tr>
<td>ipDestContainer</td>
<td>Reference To ITcVnContainer</td>
<td>Returns the resulting container</td>
</tr>
<tr>
<td>nIntDivideByZeroResult</td>
<td>LINT</td>
<td>Value set as result in case of integer division by zero</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [1 135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcContainer1</td>
<td>ITcVnContainer</td>
<td>Source container 1</td>
</tr>
<tr>
<td>ipSrcContainer2</td>
<td>ITcVnContainer</td>
<td>Source container 2</td>
</tr>
<tr>
<td>ipDestContainer</td>
<td>Reference To ITcVnContainer</td>
<td>Returns the resulting container</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.2.10 F_VN_MinContainer

Element-wise minimum of two containers (same length and type).

### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_MinContainer : HRESULT
VAR_INPUT
  ipSrcContainer1 : ITcVnContainer;
  ipSrcContainer2 : ITcVnContainer;
  ipDestContainer : Reference To ITcVnContainer;
  hrPrev : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcContainer1</td>
<td>ITcVnContainer [345]</td>
<td>Source container 1</td>
</tr>
<tr>
<td>ipSrcContainer2</td>
<td>ITcVnContainer [345]</td>
<td>Source container 2</td>
</tr>
<tr>
<td>ipDestContainer</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns the resulting container</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.2.11 F_VN_MultiplyContainers

Element-wise multiplication of two containers (same length and type).

**Syntax**

**Definition:**

```c
FUNCTION F_VN_MultiplyContainers : HRESULT
VAR_INPUT
   ipSrcContainer1 : ITcVnContainer;
   ipSrcContainer2 : ITcVnContainer;
   ipDestContainer : Reference To ITcVnContainer;
   hrPrev : HRESULT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcContainer 1</td>
<td>ITCvNContainer [345]</td>
<td>Source container 1</td>
</tr>
<tr>
<td>ipSrcContainer 2</td>
<td>ITCvNContainer [345]</td>
<td>Source container 2</td>
</tr>
<tr>
<td>ipDestContainer</td>
<td>Reference To ITCvNContainer [345]</td>
<td>Returns the resulting container</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.2.12  **F_VN_MultiplyWithContainerElements1**

Multiply each container element with a value.

**Syntax**

**Definition:**

FUNCTION F_VN_MultiplyWithContainerElements1 : HRESULT

VAR_INPUT

\[
\begin{align*}
  & fValue : \text{LREAL}; \\
  & ipContainer : \text{ITCvNContainer}; \\
  & hrPrev : \text{HRESULT};
\end{align*}
\]

END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fValue</td>
<td>LREAL</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer [345]</td>
<td>Container with 1-dimensional elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [$135$]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.2.13  F_VN_MultiplyWithContainerElements2

Multiply each container element with a value.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_MultiplyWithContainerElements2 : HRESULT
VAR_INPUT
    aValue : Reference To TcVnVector2_LREAL;
    ipContainer : ITcVnContainer;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector2_LREAL</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with 2-dimensional elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [$135$]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [$135$]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.2.14  F_VN_MultiplyWithContainerElements3

Multiply each container element with a value.

Syntax

Definition:

FUNCTION F_VN_MultiplyWithContainerElements3 : HRESULT
VAR_INPUT
  aValue : Reference To TcVnVector3_LREAL;
  ipContainer : ITcVnContainer;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector3_LREAL</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with 3-dimensional elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

Development environment | Target platform | PLC libraries to include |
------------------------|-----------------|-------------------------|
TwinCAT V3.1.4024.17 or later | PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU | Tc3_Vision

6.4.2.15  F_VN_MultiplyWithContainerElements4

Multiply each container element with a value.

Syntax

Definition:

FUNCTION F_VN_MultiplyWithContainerElements4 : HRESULT
VAR_INPUT
  aValue : Reference To TcVnVector4_LREAL;
  ipContainer : ITcVnContainer;
  hrPrev : HRESULT;
END_VAR
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>Value</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer [153]</td>
<td>Container with 4-dimensional elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.2.16 **F_VN_NegateContainer**

Element-wise negation of a container (two's complement).

**Syntax**

**Definition:**

```
FUNCTION F_VN_NegateContainer : HRESULT
VAR_INPUT
    ipSrcContainer  : ITCvNContainer;
    ipDestContainer : Reference To ITCvNContainer;
    hrPrev             : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcContainer</td>
<td>ITCvNContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>ipDestContainer</td>
<td>Reference To ITCvNContainer [345]</td>
<td>Returns the resulting container (same type as ipSrcContainer)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.2.17 **F_VN_SubtractContainers**

Element-wise subtraction of two containers (same length and type).

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcContainer1</td>
<td>ITcVnContainer [345]</td>
<td>Source container 1</td>
</tr>
<tr>
<td>ipSrcContainer2</td>
<td>ITcVnContainer [345]</td>
<td>Source container 2</td>
</tr>
<tr>
<td>ipDestContainer</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns the resulting container</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

Required License

TC3 Vision Base
### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

#### 6.4.3 Algebraic Image Operations

This group contains functions for applying algebraic operations to images pixel by pixel.

**Functions**

**Addition**
- `F_VN_AddImages` [468]
- `F_VN_AddImagesWeighted(Exp)` [469]
- `F_VN_AddScalarToImage` [471]
- `F_VN_AddVectorToImage` [472]

**Subtraction**
- `F_VN_SubtractImages` [507]
- `F_VN_SubtractImageFromScalar` [505]
- `F_VN_SubtractImageFromVector` [506]
- `F_VN_SubtractScalarFromImage` [508]
- `F_VN_SubtractVectorFromImage` [509]

**Multiplication**
- `F_VN_MultiplyImages` [503]
- `F_VN_MultiplyImageWithScalar` [503]
- `F_VN_MultiplyImageWithVector` [504]

**Division**
- `F_VN_DivideImages` [487]
- `F_VN_DivideImageByScalar` [485]
- `F_VN_DivideImageByVector` [486]
- `F_VN_DivideScalarByImage` [488]
- `F_VN_DivideVectorByImage` [489]

**Exponentiation and logarithmization**
- `F_VN_ElementwiseExp` [490]
- `F_VN_ElementwiseLog` [491]

**Bitwise AND**
- `F_VN_BitwiseAndImages(Exp)` [473]
- `F_VN_BitwiseAndScalarWithImage` [474]
- `F_VN_BitwiseAndVectorWithImage` [475]

**Bitwise NOT**
- `F_VN_BitwiseNotImage(Exp)` [476]

**Bitwise OR**
6.4.3.1 F_VN_AddImages

Element-wise addition of two images using saturation arithmetics.

Syntax

Definition:

```
FUNCTION F_VN_AddImages : HRESULT
VAR_INPUT
    ipSrcImage1 : ITcVnImage;
    ipSrcImage2 : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage1</td>
<td>ITcVnImage [483]</td>
<td>First source image</td>
</tr>
<tr>
<td>ipSrcImage2</td>
<td>ITcVnImage [483]</td>
<td>Second source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Samples

- Copy image areas [1367]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.3.2  F_VN_AddImagesWeighted

Weighted, element-wise addition of two images using saturation arithmetics.

Syntax

Definition:

FUNCTION F_VN_AddImagesWeighted : HRESULT  
VAR_INPUT  
  ipSrcImage1 : ITcVnImage;  
  ipSrcImage2 : ITcVnImage;  
  ipDestImage : Reference To ITcVnImage;  
  fWeight1 : LREAL;  
  fWeight2 : LREAL;  
  fDelta : LREAL;  
  hrPrev : HRESULT;  
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage1</td>
<td>ITcVnImage [383]</td>
<td>First source image</td>
</tr>
<tr>
<td>ipSrcImage2</td>
<td>ITcVnImage [383]</td>
<td>Second source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>fWeight1</td>
<td>LREAL</td>
<td>Weight factor applied to ipSrcImage1</td>
</tr>
<tr>
<td>fWeight2</td>
<td>LREAL</td>
<td>Weight factor applied to ipSrcImage2</td>
</tr>
<tr>
<td>fDelta</td>
<td>LREAL</td>
<td>Value added to the weighted sum of both images</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
**Return value**

HRESULT \[\text{135}\]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.3.3 \text{F\_VN\_AddImagesWeightedExp}**

Weighted, element-wise addition of two images using saturation arithmetics.

**Syntax**

**Definition:**

\[
\text{FUNCTION F\_VN\_AddImagesWeightedExp : HRESULT}\\
\text{VAR\_INPUT}\\
\quad \text{ipSrcImage1 : ITcVnImage;}\\
\quad \text{ipSrcImage2 : ITcVnImage;}\\
\quad \text{ipDestImage : Reference To ITcVnImage;}\\
\quad \text{fWeight1 : LREAL;}\\
\quad \text{fWeight2 : LREAL;}\\
\quad \text{fDelta : LREAL;}\\
\quad \text{eDestType : ETcVnElementType;}\\
\quad \text{hrPrev : HRESULT;}\\
\text{END\_VAR}
\]

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage1</td>
<td>ITcVnImage [383]</td>
<td>First source image</td>
</tr>
<tr>
<td>ipSrcImage2</td>
<td>ITcVnImage [383]</td>
<td>Second source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>fWeight1</td>
<td>LREAL</td>
<td>Weight factor applied to ipSrcImage1</td>
</tr>
<tr>
<td>fWeight2</td>
<td>LREAL</td>
<td>Weight factor applied to ipSrcImage2</td>
</tr>
<tr>
<td>fDelta</td>
<td>LREAL</td>
<td>Value added to the weighted sum of both images</td>
</tr>
<tr>
<td>eDestType</td>
<td>ETcVnElementType [186]</td>
<td>Destination image depth</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.3.4  F_VN_AddScalarToImage

Add a scalar value to each image pixel using saturation arithmetics.

Syntax

Definition:

FUNCTION F_VN_AddScalarToImage : HRESULT
VAR_INPUT
  fScalar : LREAL;
  ipSrcImage : ITCvnlmage;
  ipDestImage : Reference To ITCvnlmage;
  hrPrev : HRESULT;
END_VAR

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fScalar</td>
<td>LREAL</td>
<td>Scalar value</td>
</tr>
<tr>
<td>ipSrcImage</td>
<td>ITCvnlmage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITCvnlmage [383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License
TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.3.5 F_VN_AddVectorToImage

Add a vector (1 element for each image channel) to each image pixel using saturation arithmetics.

Syntax

Definition:

FUNCTION F_VN_AddVectorToImage : HRESULT
VAR_INPUT
  aVector : Reference To TcVnVector4_LREAL;
  ipSrcImage : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)</td>
</tr>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.3.6  F_VN_BitwiseAndImages

**F_VN_BitwiseAndImages**

- **ipSrcImage1**  ITcVnImage
- **ipSrcImage2**  ITcVnImage
- **ipDestImage**  Reference To ITcVnImage
- **hrPrev**  HRESULT

Element-wise application of a bit-wise AND operator to two images.

**Syntax**

**Definition:**

FUNCTION F_VN_BitwiseAndImages : HRESULT
VAR_INPUT
  ipSrcImage1 : ITcVnImage;
  ipSrcImage2 : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  hrPrev : HRESULT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage1</td>
<td>ITcVnImage [383]</td>
<td>First source image</td>
</tr>
<tr>
<td>ipSrcImage2</td>
<td>ITcVnImage [383]</td>
<td>Second source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.3.7  F_VN_BitwiseAndImagesExp

**F_VN_BitwiseAndImagesExp**

- **ipSrcImage1**  ITcVnImage
- **ipSrcImage2**  ITcVnImage
- **ipDestImage**  Reference To ITcVnImage
- **ipMask**  ITcVnImage
- **hrPrev**  HRESULT

Element-wise application of a bit-wise AND operator to two images. (expert function)
Syntax

Definition:

FUNCTION F_VN_BitwiseAndImagesExp : HRESULT
VAR_INPUT
  ipSrcImage1 : ITcVnImage;
  ipSrcImage2 : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  ipMask     : ITcVnImage;
  hrPrev     : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage1</td>
<td>ITcVnImage [135]</td>
<td>First source image</td>
</tr>
<tr>
<td>ipSrcImage2</td>
<td>ITcVnImage [135]</td>
<td>Second source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [135]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITcVnImage [135]</td>
<td>Mask to limit the operation to specific pixel positions (mask value 0: skip pixel, mask value &gt; 0: apply operation to pixel)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.3.8 F_VN_BitwiseAndScalarWithImage

Bitwise and a scalar value with each image pixel.

Syntax

Definition:

FUNCTION F_VN_BitwiseAndScalarWithImage : HRESULT
VAR_INPUT
  fScalar      : LREAL;
  ipSrcImage   : ITcVnImage;
  ipDestImage  : Reference To ITcVnImage;
  hrPrev       : HRESULT;
END_VAR
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fScalar</td>
<td>LREAL</td>
<td>Scalar value</td>
</tr>
<tr>
<td>ipSrcImage</td>
<td>ITCvNImage</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITCvNImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [②135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.3.9 F_VN_BitwiseAndVectorWithImage

Bitwise and a vector (1 element for each image channel) with each image pixel.

Syntax

Definition:

```c
FUNCTION F_VN_BitwiseAndVectorWithImage : HRESULT
VAR_INPUT
  aVector : Reference To TcVnVector4_LREAL;
  ipSrcImage : ITCvNImage;
  ipDestImage : Reference To ITCvNImage;
  hrPrev : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)</td>
</tr>
<tr>
<td>ipSrcImage</td>
<td>ITCvNImage</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITCvNImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT ⏤ 135

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.3.10 F_VN_BitwiseNotImage

Element-wise application of a bit-wise NOT operator to an image.

Syntax

Definition:

FUNCTION F_VN_BitwiseNotImage : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage[138]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage[138]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT ⏤ 135

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.3.11  F_VN_BitwiseNotImageExp

Element-wise application of a bit-wise NOT operator to an image. (expert function)

Syntax

Definition:

FUNCTION F_VN_BitwiseNotImageExp : HRESULT
VAR_INPUT
   ipSrcImage : ITCvImage;
   ipDestImage : Reference To ITCvImage;
   ipMask : ITCvImage;
   hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvImage</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITCvImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITCvImage</td>
<td>Mask to limit the operation to specific pixel positions (mask value 0: skip pixel, mask value &gt; 0: apply operation to pixel)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.3.12  F_VN_BitwiseOrImages

Element-wise application of a bit-wise OR operator to two images.
Syntax

Definition:

FUNCTION F_VN_BitwiseOrImages : HRESULT
VAR_INPUT
  ipSrcImage1 : ITcVnImage;
  ipSrcImage2 : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage1</td>
<td>ITcVnImage</td>
<td>First source image</td>
</tr>
<tr>
<td>ipSrcImage2</td>
<td>ITcVnImage</td>
<td>Second source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.3.13 F_VN_BitwiseOrImagesExp

Element-wise application of a bit-wise OR operator to two images. (expert function)

Syntax

Definition:

FUNCTION F_VN_BitwiseOrImagesExp : HRESULT
VAR_INPUT
  ipSrcImage1 : ITcVnImage;
  ipSrcImage2 : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  ipMask : ITcVnImage;
  hrPrev : HRESULT;
END_VAR
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage1</td>
<td>ITcVnImage [383]</td>
<td>First source image</td>
</tr>
<tr>
<td>ipSrcImage2</td>
<td>ITcVnImage [383]</td>
<td>Second source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITcVnImage [383]</td>
<td>Mask to limit the operation to specific pixel positions (mask value 0: skip pixel, mask value &gt; 0: apply operation to pixel)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

**HRESULT [135]**

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.3.14 F_VN_BitwiseOrScalarWithImage

```
FUNCTION F_VN_BitwiseOrScalarWithImage : HRESULT
VAR_INPUT
  fScalar : LREAL;
  ipSrcImage : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  hrPrev : HRESULT;
END_VAR
```

Bitwise or a scalar value with each image pixel.

**Syntax**

**Definition:**

```
FUNCTION F_VN_BitwiseOrScalarWithImage : HRESULT
VAR_INPUT
  fScalar : LREAL;
  ipSrcImage : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  hrPrev : HRESULT;
END_VAR
```
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fScalar</td>
<td>LREAL</td>
<td>Scalar value</td>
</tr>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.3.15 F_VN_BitwiseOrVectorWithImage

Bitwise or a vector (1 element for each image channel) with each image pixel.

Syntax

Definition:

```
FUNCTION F_VN_BitwiseOrVectorWithImage : HRESULT
    VAR_INPUT
        aVector : Reference To TcVnVector4_LREAL;
        ipSrcImage : ITcVnImage;
        ipDestImage : Reference To ITcVnImage;
        hrPrev : HRESULT;
    END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)</td>
</tr>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.3.16 F_VN_BitwiseXorImages

Element-wise application of a bit-wise XOR operator to two images.

Syntax

Definition:

FUNCTION F_VN_BitwiseXorImages : HRESULT
VAR_INPUT
   ipSrcImage1 : ITCnImage;
   ipSrcImage2 : ITCnImage;
   ipDestImage : Reference To ITCnImage;
   hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage1</td>
<td>ITCnImage [383]</td>
<td>First source image</td>
</tr>
<tr>
<td>ipSrcImage2</td>
<td>ITCnImage [383]</td>
<td>Second source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITCnImage [383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.3.17 F_VN_BitwiseXorImagesExp

Element-wise application of a bit-wise XOR operator to two images. (expert function)

Syntax

```c
FUNCTION F_VN_BitwiseXorImagesExp : HRESULT
VAR_INPUT
    ipSrcImage1 : ITcVnImage;
    ipSrcImage2 : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    ipMask      : ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage1</td>
<td>ITcVnImage</td>
<td>First source image</td>
</tr>
<tr>
<td>ipSrcImage2</td>
<td>ITcVnImage</td>
<td>Second source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITcVnImage</td>
<td>Mask to limit the operation to specific pixel positions (mask value 0: skip pixel, mask value &gt; 0: apply operation to pixel)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

```
HRESULT [135]
```

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
### 6.4.3.18  F_VN_BitwiseXorScalarWithImage

#### Definition:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F_VN_BitwiseXorScalarWithImage</td>
<td>Bitwise xor a scalar value with each image pixel.</td>
</tr>
</tbody>
</table>

#### Syntax

**Definition:**

```c
FUNCTION F_VN_BitwiseXorScalarWithImage : HRESULT
VAR_INPUT
  fScalar : LREAL;
  ipSrcImage : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  hrPrev : HRESULT;
END_VAR
```

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fScalar</td>
<td>LREAL</td>
<td>Scalar value</td>
</tr>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

#### Return value

- HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.3.19  F_VN_BitwiseXorVectorWithImage

#### Definition:

**Definition:**

```c
FUNCTION F_VN_BitwiseXorVectorWithImage : HRESULT
VAR_INPUT
  aVector : Reference To TcVnVector4_LREAL;
  ipSrcImage : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  hrPrev : HRESULT;
END_VAR
```

##### Bitwise xor a vector (1 element for each image channel) with each image pixel.

#### Syntax

**Definition:**

```c
FUNCTION F_VN_BitwiseXorVectorWithImage : HRESULT
VAR_INPUT
  aVector : Reference To TcVnVector4_LREAL;
  ipSrcImage : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  hrPrev : HRESULT;
END_VAR
```
FUNCTION F_VN_BitwiseXorVectorWithImage : HRESULT
VAR_INPUT
  aVector : Reference To TcVnVector4_LREAL;
  ipSrcImage : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  hrPrev : HRESULT;
END_VAR

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)</td>
</tr>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [1]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [2]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [3]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.3.20 **F_VN_BlendImages**

Blends two images, i.e. dest = (src1 * weight1 + src2 * weight2) / (weight1 + weight2).
Can use available TwinCAT Job Tasks for executing parallel code regions.

### Syntax

**Definition:**

FUNCTION F_VN_BlendImages : HRESULT
VAR_INPUT
  ipSrcImage1 : ITcVnImage;
  ipSrcImage2 : ITcVnImage;
  ipWeights1 : ITcVnImage;
  ipWeights2 : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  hrPrev : HRESULT;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage1</td>
<td>ITcVnImage</td>
<td>First source image (ET_USINT or ET_REAL)</td>
</tr>
<tr>
<td>ipSrcImage2</td>
<td>ITcVnImage</td>
<td>Second source image (same type and size as ipSrcImage1)</td>
</tr>
<tr>
<td>ipWeights1</td>
<td>ITcVnImage</td>
<td>Weights for ipSrcImage1 (1 channel, ET_REAL)</td>
</tr>
<tr>
<td>ipWeights2</td>
<td>ITcVnImage</td>
<td>Weights for ipSrcImage2 (1 channel, ET_REAL)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.3.21 F_VN_DivideImageByScalar

**F_VN_DivideImageByScalar**

- **ipSrcImage** ITcVnImage
- **fScalar** LREAL
- **ipDestImage** Reference To ITcVnImage
- **hrPrev** HRESULT

Divide each image pixel by a scalar value using saturation arithmetics.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_DivideImageByScalar : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    fScalar    : LREAL;
    ipDestImage : Reference To ITcVnImage;
    hrPrev     : HRESULT;
END_VAR
```
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage[383]</td>
<td>Source image</td>
</tr>
<tr>
<td>fScalar</td>
<td>LREAL</td>
<td>Scalar value</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage[383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT[135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.3.22 F_VN_DivideImageByVector

Divides each image pixel by a vector (1 element for each image channel) using saturation arithmetics.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_DivideImageByVector : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    aVector : Reference To TcVnVector4_LREAL;
    ipDestImage : Reference To ITcVnImage;
    hrPrev : HRESULT;
END_VAR
```

F_VN_DivideImageByVector

```c
HRESULT F_VN_DivideImageByVector
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.3.23 F_VN_DivideImages

F_VN_DivideImages

**Syntax**

**Definition:**

FUNCTION F_VN_DivideImages : HRESULT
VAR_INPUT
  ipSrcImage1 : ITcVnImage;
  ipSrcImage2 : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  hrPrev : HRESULT;
END_VAR

Element-wise division of two images using saturation arithmetics. (A division by zero equals zero.)
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage1</td>
<td>ITcVnImage [383]</td>
<td>First source image (dividend)</td>
</tr>
<tr>
<td>ipSrcImage2</td>
<td>ITcVnImage [383]</td>
<td>Second source image (divisor)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Further information

The function F_VN_DivideImages returns zero for the respective elements in the case of division by zero. However, this only works if the option Floating point exceptions of the executing PLC task [58] is deactivated. Otherwise a division-by-zero error occurs.

NOTE

Floating point exceptions

This function can unnecessarily cause errors if the option Floating point exceptions of the executing PLC task [58] is active. Therefore, deactivate this option.

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.3.24 F_VN_DivideScalarByImage

Divides a scalar value by each image pixel using saturation arithmetics.

Syntax

Definition:

FUNCTION F_VN_DivideScalarByImage : HRESULT
VAR_INPUT
  fScalar     : LREAL;
  ipSrcImage  : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  hrPrev      : HRESULT;
END_VAR
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fScalar</td>
<td>LREAL</td>
<td>Scalar value</td>
</tr>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.3.25 F_VN_DivideVectorByImage

**Definition:**

```c
FUNCTION F_VN_DivideVectorByImage : HRESULT
VAR_INPUT
  aVector : Reference To TcVnVector4_LREAL;
  ipSrcImage : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  hrPrev : HRESULT;
END_VAR
```

Divides a vector (1 element for each image channel) by each image pixel using saturation arithmetics.

**Syntax**
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4 LREAL</td>
<td>4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)</td>
</tr>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.3.26 F_VN_ElementwiseExp**

Computes the natural exponent of each pixel value.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_ElementwiseExp : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image (REAL or LREAL)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (Same type and size as ipSrcImage, an appropriate image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.3.27 F_VN_ElementwiseLog**

**Syntax**

**Definition:**

FUNCTION F_VN_ElementwiseLog : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  hrPrev : HRESULT;
END_VAR

Comes the natural logarithm of each pixel value.

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image (REAL or LREAL)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (Same type and size as ipSrcImage, an appropriate image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.3.28  F_VN_MaxImage

Element-wise maximum of two images.

Syntax

Definition:

```
FUNCTION F_VN_MaxImage : HRESULT
VAR_INPUT
    ipSrcImage1 : ITcVnImage;
    ipSrcImage2 : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev     : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage1</td>
<td>ITcVnImage</td>
<td>First source image</td>
</tr>
<tr>
<td>ipSrcImage2</td>
<td>ITcVnImage</td>
<td>Second source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.3.29  F_VN_MaxImageExp

Element-wise maximum of two images. (expert function)
Syntax

Definition:

FUNCTION F_VN_MaxImageExp : HRESULT
VAR_INPUT
 ipSrcImage1 : ITcVnImage;
 ipSrcImage2 : ITcVnImage;
 ipDestImage : Reference To ITcVnImage;
 ipMask : ITcVnImage;
 hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage1</td>
<td>ITcVnImage [383]</td>
<td>First source image</td>
</tr>
<tr>
<td>ipSrcImage2</td>
<td>ITcVnImage [383]</td>
<td>Second source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITcVnImage [383]</td>
<td>Mask of type USINT</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.3.30 F_VN_MaxImageWithScalar

F_VN_MaxImageWithScalar

fScalar LREAL HRESULT F_VN_MaxImageWithScalar

ipSrcImage ITcVnImage

ipDestImage Reference To ITcVnImage

hrPrev HRESULT

Element-wise maximum of image and scalar value.

Syntax

Definition:

FUNCTION F_VN_MaxImageWithScalar : HRESULT
VAR_INPUT
 fScalar : LREAL;
 ipSrcImage : ITcVnImage;
 ipDestImage : Reference To ITcVnImage;
 hrPrev : HRESULT;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fScalar</td>
<td>LREAL</td>
<td>Scalar value</td>
</tr>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.3.31 F_VN_MaxImageWithScalarExp

Element-wise maximum of image and scalar value. (expert function)

#### Syntax

**Function Definition:**

```plaintext
FUNCTION F_VN_MaxImageWithScalarExp : HRESULT
VAR_INPUT
  fScalar : LREAL;
  ipSrcImage : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  ipMask : ITcVnImage;
  hrPrev : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fScalar</td>
<td>LREAL</td>
<td>Scalar value</td>
</tr>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>383</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>383</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITcVnImage</td>
<td>383</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>135</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.32 F_VN_MaxImageWithVector

Element-wise maximum of image and vector (1 element for each image channel).

#### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_MaxImageWithVector : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector4_LREAL;
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev : HRESULT;
END_VAR
```

```plaintext
F_VN_MaxImageWithVector
    aVector Reference To TcVnVector4_LREAL ,
    ipSrcImage ITcVnImage,
    ipDestImage Reference To ITcVnImage,
    hrPrev HRESULT
```

The function `F_VN_MaxImageWithVector` performs the element-wise maximum of the source image and the vector, resulting in a new image where each channel is the maximum value of the corresponding channel in the source image and the vector.
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)</td>
</tr>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.3.33 F_VN_MaxImageWithVectorExp

Element-wise maximum of image and vector (1 element for each image channel). (expert function)

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_MaxImageWithVectorExp : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector4_LREAL;
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    ipMask : ITcVnImage;
    hrPrev : HRESULT;
END_VAR
```
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_REAL [153]</td>
<td>4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)</td>
</tr>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITcVnImage [383]</td>
<td>Mask of type USINT</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.3.34 F_VN_MinImage

Element-wise minimum of two images.

Syntax

Definition:

FUNCTION F_VN_MinImage : HRESULT
VAR_INPUT
  ipSrcImage1 : ITcVnImage;
  ipSrcImage2 : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  hrPrev : HRESULT;
END_VAR
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage1</td>
<td>ITcVnImage [383]</td>
<td>First source image</td>
</tr>
<tr>
<td>ipSrcImage2</td>
<td>ITcVnImage [383]</td>
<td>Second source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.3.35 F_VN_MinImageExp**

Element-wise minimum of two images. (expert function)

**Syntax**

**Definition:**

FUNCTION F_VN_MinImageExp : HRESULT
VAR_INPUT
  ipSrcImage1 : ITcVnImage;
  ipSrcImage2 : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  ipMask : ITcVnImage;
  hrPrev : HRESULT;
END_VAR
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage1</td>
<td>ITcVnImage</td>
<td>First source image</td>
</tr>
<tr>
<td>ipSrcImage2</td>
<td>ITcVnImage</td>
<td>Second source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITcVnImage</td>
<td>Mask of type USINT</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.3.36 **F_VN_MinImageWithScalar**

**Syntax**

**Definition:**

```func
FUNCTION F_VN_MinImageWithScalar : HRESULT
VAR_INPUT
  fScalar : LREAL;
  ipSrcImage : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  hrPrev : HRESULT;
END_VAR
HRESULT F_VN_MinImageWithScalar
```

Element-wise minimum of image and scalar value.

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fScalar</td>
<td>LREAL</td>
<td>Scalar value</td>
</tr>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_vision</td>
</tr>
</tbody>
</table>

#### 6.4.3.37 F_VN_MinImageWithScalarExp

**F_VN_MinImageWithScalarExp**

- **fScalar** : LREAL
- **ipSrcImage** : ITcVnImage
- **ipDestImage** : Reference To ITcVnImage
- **ipMask** : ITcVnImage
- **hrPrev** : HRESULT

Element-wise minimum of image and scalar value. (expert function)

**Syntax**

**Definition:**

```c
FUNCTION F_VN_MinImageWithScalarExp : HRESULT
VAR_INPUT
  fScalar : LREAL;
  ipSrcImage : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  ipMask : ITcVnImage;
  hrPrev : HRESULT;
END_VAR

RETURN HRESULT F_VN_MinImageWithScalarExp
```

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fScalar</td>
<td>LREAL</td>
<td>Scalar value</td>
</tr>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITcVnImage [383]</td>
<td>Mask of type USINT</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

#### Return value

HRESULT [135]

### Required License

TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.3.38 F_VN_MinImageWithVector

Element-wise minimum of image and vector (1 element for each image channel).

Syntax

Definition:

```
FUNCTION F_VN_MinImageWithVector : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector4_LREAL;
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)</td>
</tr>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

```
HRESULT [135]
```

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.3.39 F_VN_MinImageWithVectorExp

**F_VN_MinImageWithVectorExp**

- **aVector**: Reference To TcVnVector4_LREAL
- **ipSrcImage**: ITCvNImage
- **ipDestImage**: Reference To ITCvNImage
- **ipMask**: ITCvNImage
- **hrPrev**: HRESULT

Element-wise minimum of image and vector (1 element for each image channel). (expert function)

**Syntax**

Definition:

```c
FUNCTION F_VN_MinImageWithVectorExp : HRESULT
VAR_INPUT
aVector : Reference To TcVnVector4_LREAL;
ipSrcImage : ITCvNImage;
ipDestImage : Reference To ITCvNImage;
ipMask : ITCvNImage;
hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)</td>
</tr>
<tr>
<td>ipSrcImage</td>
<td>ITCvNImage</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITCvNImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITCvNImage</td>
<td>Mask of type USINT</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
### 6.4.3.40 F_VN_MultiplyImages

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F_VN_MultiplyImages</td>
<td>Element-wise multiplication of two images using saturation arithmetics.</td>
</tr>
</tbody>
</table>

**Syntax**

**Definition:**

```
FUNCTION F_VN_MultiplyImages : HRESULT
VAR_INPUT
  ipSrcImage1 : ITcVnImage;
  ipSrcImage2 : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage1</td>
<td>ITcVnImage [1383]</td>
<td>First source image</td>
</tr>
<tr>
<td>ipSrcImage2</td>
<td>ITcVnImage [1383]</td>
<td>Second source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [1383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.3.41 F_VN_MultiplyImageWithScalar

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F_VN_MultiplyImageWithScalar</td>
<td>Multiply each image pixel by a scalar value using saturation arithmetics.</td>
</tr>
</tbody>
</table>

**Syntax**

**Definition:**

```
FUNCTION F_VN_MultiplyImageWithScalar : HRESULT
VAR_INPUT
  fScalar : LREAL;
  ipSrcImage : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  hrPrev : HRESULT;
END_VAR
```
**Syntax**

**Definition:**

```c
FUNCTION F_VN_MultiplyImageWithScalar : HRESULT
VAR_INPUT
    fScalar : LREAL;
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fScalar</td>
<td>LREAL</td>
<td>Scalar value</td>
</tr>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage {383}</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT {135}</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT {135}

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.3.42 **F_VN_MultiplyImageWithVector**

Multiply each image pixel by a vector (1 element for each image channel) using saturation arithmetics.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_MultiplyImageWithVector : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector4_LREAL;
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev : HRESULT;
END_VAR
```
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4 LREAL [153]</td>
<td>4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)</td>
</tr>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

#### 6.4.3.43 F_VN_SubtractImageFromScalar

Subtract each image pixel from a scalar value using saturation arithmetics.

**Syntax**

**Definition:**

```
FUNCTION F_VN_SubtractImageFromScalar : HRESULT
VAR_INPUT
  fScalar : LREAL;
  ipSrcImage : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  hrPrev : HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fScalar</td>
<td>LREAL</td>
<td>Scalar value</td>
</tr>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage[383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage[383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.3.44 F_VN_SubtractImageFromVector

\[
\text{F}_\text{VN}_\text{SubtractImageFromVector} \quad \text{aVector} \quad \text{ipSrcImage} \quad \text{ipDestImage} \quad \text{hrPrev}
\]

Subtract each image pixel from a vector (1 element for each image channel) using saturation arithmetics.

### Syntax

**Definition:**

```c
FUNCTION F_VN_SubtractImageFromVector : HRESULT
VAR_INPUT
  aVector : Reference To TcVnVector4_LREAL;
  ipSrcImage : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  hrPrev : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aVector</td>
<td>Reference To TcVnVector4 LREAL</td>
<td>4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)</td>
</tr>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.3.45 F_VN_SubtractImages

Element-wise subtraction of two images using saturation arithmetics.

### Syntax

**Definition:**

```
FUNCTION F_VN_SubtractImages : HRESULT
VAR_INPUT
    ipSrcImage1 : ITcVnImage;
    ipSrcImage2 : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev : HRESULT;
END_VAR
```
# Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage1</td>
<td>ITcVnImage ![383]</td>
<td>First source image (minuend)</td>
</tr>
<tr>
<td>ipSrcImage2</td>
<td>ITcVnImage ![383]</td>
<td>Second source image (subtrahend)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage ![383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT ![135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

# Return value

HRESULT ![135]

# Required License

TC3 Vision Base

# System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.3.46 F_VN_SubtractScalarFromImage

![F_VN_SubtractScalarFromImage]

Subtract a scalar value from each image pixel using saturation arithmetics.

### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_SubtractScalarFromImage : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  fScalar : LREAL;
  ipDestImage : Reference To ITcVnImage;
  hrPrev : HRESULT;
END_VAR
```

# Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage ![383]</td>
<td>Source image</td>
</tr>
<tr>
<td>fScalar</td>
<td>LREAL</td>
<td>Scalar value</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage ![383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT ![135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.3.47  F_VN_SubtractVectorFromImage

Subtract a vector (1 element for each image channel) from each image pixel using saturation arithmetics.

Syntax

Definition:

FUNCTION F_VN_SubtractVectorFromImage : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  aVector : Reference To TcVnVector4_LREAL;
  ipDestImage : Reference To ITcVnImage;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage[383]</td>
<td>Source image</td>
</tr>
<tr>
<td>aVector</td>
<td>Reference To</td>
<td>4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)</td>
</tr>
<tr>
<td></td>
<td>TcVnVector4_LREAL[153]</td>
<td></td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td></td>
<td>ITcVnImage[383]</td>
<td></td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License
TC3 Vision Base
**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4 Basic Container Operations

This group contains functions for handling containers [143].

**Functions**

**Creation of containers**

- F_VN_CopyContainer  [690]
- F_VN_CopyContainerElementsConditional_ITcVnContainer  [690]
- F_VN_CopyContainerElementsConditional_ITcVnForwardIterator  [691]
- F_VN_CreateContainer  [693]
- F_VN_CreateContainerFromArray  [694]
- F_VN_ExtractContainerRange  [703]
- F_VN_ReserveContainerMemory  [710]

**Handling of container elements**

- F_VN_AppendToContainer  [511]
- F_VN_CheckIfEmpty  [686]
- F_VN_ConvertContainerType  [688]
- F_VN_EraseFromContainer  [695]
- F_VN_FillContainer(Exp)  [564]
- F_VN_GetAt  [597]
- F_VN_GetContainer(Exp)  [703]
- F_VN_GetNumberOfElements  [706]
- F_VN_InsertIntoContainer  [626]
- F_VN_ReverseContainer  [711]
- F_VN_SetAt  [656]
- F_VN_SetContainer  [712]

**Exporting container data to the memory area**

- F_VN_ExportContainer  [696]
- F_VN_ExportSubContainer  [699]
- F_VN_ExportContainer_String  [697]
- F_VN_ExportSubContainer_String  [700]
- F_VN_ExportContainerSize  [698]
- F_VN_ExportSubContainerSize  [702]

**Handling of iterators**

- F_VN_AdvanceIterator  [585]
- F_VN_CheckIfIteratorIsAtEnd  [687]
6.4.4.1  F_VN_AppendToContainer

This chapter contains functions to add an element to a container according to the data type.

6.4.4.1.1  F_VN_AppendToContainer_DINT

Append a single element of type DINT to a container.

Syntax

Definition:
FUNCTION F_VN_AppendToContainer_DINT : HRESULT
VAR_INPUT
   nElement : DINT;
   ipContainer : ITcVnContainer;
   hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nElement</td>
<td>DINT</td>
<td>Single element to append to ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container to which the element will be appended</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.4.1.2  F_VN_AppendToContainer_INT

**F_VN_AppendToContainer_INT**

- **nElement**  INT  
- **ipContainer**  ITcVnContainer  
- **hrPrev**  HRESULT

Append a single element of type INT to a container.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_AppendToContainer_INT : HRESULT
VAR_INPUT
    nElement    : INT;
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nElement</td>
<td>INT</td>
<td>Single element to append to ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container to which the element will be appended</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.1.3  F_VN_AppendToContainer_ITcVnContainer

**F_VN_AppendToContainer_ITcVnContainer**

- **ipElement**  ITcVnContainer  
- **ipContainer**  ITcVnContainer  
- **hrPrev**  HRESULT

Append a single element to a container or concatenate containers (if ipElement has the same typeId as ipContainer).

**Syntax**

**Definition:**

```c
FUNCTION F_VN_AppendToContainer_ITcVnContainer : HRESULT
VAR_INPUT
    ipElement : ITcVnContainer;
```
F_VN_AppendToContainer_ITcVnForwardIterator

Append a single element (represented by an iterator) to a container.

**Syntax**

**Definition:**

FUNCTION F_VN_AppendToContainer_ITcVnForwardIterator : HRESULT

VAR_INPUT

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipElement</td>
<td>ITCvNContainer [345]</td>
<td>Single element to append to ipContainer or container with several elements to be concatenated</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer [345]</td>
<td>Container to which the element(s) will be appended</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
## Return value

**HRESULT [135]**

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.4.1.5 F_VN_AppendToContainer_LREAL

The function `F_VN_AppendToContainer_LREAL` appends a single element of type `LREAL` to a container.

### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_AppendToContainer_LREAL : HRESULT
VAR_INPUT
    fElement : LREAL;
    ipContainer : ITcVnContainer;
    hrPrev : HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fElement</td>
<td>LREAL</td>
<td>Single element to append to ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container to which the element will be appended</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

**HRESULT [135]**

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
### 6.4.4.1.6 F_VN_AppendToContainer_REAL

**F_VN_AppendToContainer_REAL**

<table>
<thead>
<tr>
<th>fElement</th>
<th>REAL</th>
<th>HRESULT</th>
<th>F_VN_AppendToContainer_REAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td></td>
<td>F_VN_AppendToContainer_REAL</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td></td>
<td>F_VN_AppendToContainer_REAL</td>
</tr>
</tbody>
</table>

Append a single element of type REAL to a container.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_AppendToContainer_REAL : HRESULT
VAR_INPUT
   fElement : REAL;
   ipContainer : ITcVnContainer;
   hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fElement</td>
<td>REAL</td>
<td>Single element to append to ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container to which the element will be appended</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

**HRESULT [135]**

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.1.7 F_VN_AppendToContainer_SINT

**F_VN_AppendToContainer_SINT**

<table>
<thead>
<tr>
<th>nElement</th>
<th>SINT</th>
<th>HRESULT</th>
<th>F_VN_AppendToContainer_SINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td></td>
<td>F_VN_AppendToContainer_SINT</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td></td>
<td>F_VN_AppendToContainer_SINT</td>
</tr>
</tbody>
</table>

Append a single element of type SINT to a container.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_AppendToContainer_SINT : HRESULT
VAR_INPUT
   nElement : SINT;
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nElement</td>
<td>SINT</td>
<td>Single element to append to ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container to which the element will be appended</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Append a single element of type TcVnDMatch to a container.

Syntax

**Definition:**

```c
FUNCTION F_VN_AppendToContainer_TcVnDMatch : HRESULT
VAR_INPUT
    stElement : Reference To TcVnDMatch;
    ipContainer : ITCvNContainer;
    hrPrev     : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stElement</td>
<td>Reference To TcVnDMatch</td>
<td>Single element to append to ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer</td>
<td>Container to which the element will be appended</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.1.9 F_VN_AppendToContainer_TcVnKeyPoint

```c
F_VN_AppendToContainer_TcVnKeyPoint
```

- `stElement` : Reference To TcVnKeyPoint
- `ipContainer` : ITCvContainer
- `hrPrev` : HRESULT

Append a single element of type TcVnKeyPoint to a container.

**Syntax**

**Definition:**

```
FUNCTION F_VN_AppendToContainer_TcVnKeyPoint : HRESULT
VAR_INPUT
    stElement : Reference To TcVnKeyPoint;
    ipContainer : ITCvContainer;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stElement</td>
<td>Reference To TcVnKeyPoint [208]</td>
<td>Single element to append to ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvContainer [345]</td>
<td>Container to which the element will be appended</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
Append a single element of type TcVnPoint2_DINT to a container.

**Syntax**

**Definition:**

```
FUNCTION F_VN_AppendToContainer_TcVnPoint2_DINT : HRESULT
VAR_INPUT
  aElement : Reference To TcVnPoint2_DINT;
  ipContainer : ITcVnContainer;
  hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnPoint2_DINT</td>
<td>Single element to append to ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container to which the element will be appended</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

```
HRESULT [135]
```

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

Append a single element of type TcVnPoint2_LREAL to a container.

**Syntax**

**Definition:**

```
FUNCTION F_VN_AppendToContainer_TcVnPoint2_LREAL : HRESULT
VAR_INPUT
  aElement : Reference To TcVnPoint2_LREAL;
  ipContainer : ITcVnContainer;
  hrPrev : HRESULT;
END_VAR
```

Append a single element of type TcVnPoint2_LREAL to a container.
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnPoint2_REAL [151]</td>
<td>Single element to append to ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container to which the element will be appended</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

---

### Return value

**HRESULT [135]**

---

### Required License

TC3 Vision Base

---

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

---

### 6.4.4.1.12 F_VN_AppendToContainer_TcVnPoint2_REAL

`F_VN_AppendToContainer_TcVnPoint2_REAL`  

```c
FUNCTION F_VN_AppendToContainer_TcVnPoint2_REAL : HRESULT  
VAR_INPUT  
aElement    : Reference To TcVnPoint2_REAL;  
ipContainer  : ITcVnContainer;  
hrPrev       : HRESULT;  
END_VAR
```

Append a single element of type TcVnPoint2_REAL to a container.

---

### Syntax

**Definition:**

FUNCTION F_VN_AppendToContainer_TcVnPoint2_REAL : HRESULT

```c
VAR_INPUT  
aElement    : Reference To TcVnPoint2_REAL;  
ipContainer  : ITcVnContainer;  
hrPrev       : HRESULT;  
END_VAR
```
**Return value**

**HRESULT [135]**

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

8.4.4.1.13  **F_VN_AppendToContainer_TcVnPoint3_LREAL**

Append a single element of type TcVnPoint3_LREAL to a container.

**Syntax**

**Definition:**

```
FUNCTION F_VN_AppendToContainer_TcVnPoint3_LREAL : HRESULT
VAR_INPUT
    aElement    : Reference To TcVnPoint3_LREAL;
    ipContainer : ITCvnContainer;
    hrPrev      : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnPoint3_LREAL</td>
<td>Single element to append to ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvnContainer [345]</td>
<td>Container to which the element will be appended</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

**HRESULT [135]**

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
### 6.4.4.1.14 F_VN_AppendToContainer_TcVnPoint3_REAL

**F_VN_AppendToContainer_TcVnPoint3_REAL**

```plaintext
FUNCTION F_VN_AppendToContainer_TcVnPoint3_REAL : HRESULT
VAR_INPUT
  aElement : Reference To TcVnPoint3_REAL;
  ipContainer : ITCvContainer;
  hrPrev : HRESULT;
END_VAR
```

Append a single element of type TcVnPoint3_REAL to a container.

**Syntax**

**Definition:**

FUNCTION F_VN_AppendToContainer_TcVnPoint3_REAL : HRESULT
VAR_INPUT
  aElement : Reference To TcVnPoint3_REAL;
  ipContainer : ITCvContainer;
  hrPrev : HRESULT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnPoint3_REAL</td>
<td>Single element to append to ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvContainer</td>
<td>Container to which the element will be appended</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.1.15 F_VN_AppendToContainer_TcVnRectangle_DINT

**F_VN_AppendToContainer_TcVnRectangle_DINT**

```plaintext
FUNCTION F_VN_AppendToContainer_TcVnRectangle_DINT : HRESULT
VAR_INPUT
  stElement : Reference To TcVnRectangle_DINT;
  ipContainer : ITCvContainer;
  hrPrev : HRESULT;
END_VAR
```

Append a single element of type TcVnRectangle_DINT to a container.

**Syntax**

**Definition:**

FUNCTION F_VN_AppendToContainer_TcVnRectangle_DINT : HRESULT
VAR_INPUT
  stElement : Reference To TcVnRectangle_DINT;
  ipContainer : ITCvContainer;
  hrPrev : HRESULT;
END_VAR
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stElement</td>
<td>Reference To TcVnRectangle_DINT [221]</td>
<td>Single element to append to ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container to which the element will be appended</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.1.16 F_VN_AppendToContainer_TcVnVector2_DINT

Append a single element of type TcVnVector2_DINT to a container.

### Syntax

**Definition:**

```
FUNCTION F_VN_AppendToContainer_TcVnVector2_DINT : HRESULT
VAR_INPUT
    aElement : Reference To TcVnVector2_DINT;
    ipContainer : ITcVnContainer;
    hrPrev : HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector2_DINT [153]</td>
<td>Single element to append to ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container to which the element will be appended</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]
API reference

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.1.17  F_VN_AppendToContainer_TcVnVector2_INT

Append a single element of type TcVnVector2_INT to a container.

Syntax

**Definition:**

```
FUNCTION F_VN_AppendToContainer_TcVnVector2_INT : HRESULT
VAR_INPUT
    aElement : Reference To TcVnVector2_INT;
    ipContainer : ITCvnContainer;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector2_INT</td>
<td>Single element to append to ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvnContainer</td>
<td>Container to which the element will be appended</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
### F_VN_AppendToContainer_TcVnVector2_REAL

Append a single element of type TcVnVector2_REAL to a container.

#### Syntax

**Definition:**

```c
FUNCTION F_VN_AppendToContainer_TcVnVector2_REAL : HRESULT
VAR_INPUT
    aElement : Reference To TcVnVector2_REAL;
    ipContainer : ITCvnContainer;
    hrPrev : HRESULT;
END_VAR
```

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector2_REAL</td>
<td>Single element to append to ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvnContainer</td>
<td>Container to which the element will be appended</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

#### Return value

**HRESULT [135]**

#### Required License

TC3 Vision Base

#### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### F_VN_AppendToContainer_TcVnVector2_SINT

Append a single element of type TcVnVector2_SINT to a container.

#### Syntax

**Definition:**

```c
FUNCTION F_VN_AppendToContainer_TcVnVector2_SINT : HRESULT
VAR_INPUT
    aElement : Reference To TcVnVector2_SINT;
    ipContainer : ITCvnContainer;
    hrPrev : HRESULT;
END_VAR
```

524 Version: 1.3 TF7000 - TF7300
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector2_UINT [153]</td>
<td>Single element to append to ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer [345]</td>
<td>Container to which the element will be appended</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### Syntax

**Definition:**

```
FUNCTION F_VN_AppendToContainer_TcVnVector2_UINT : HRESULT
VAR_INPUT
  aElement : Reference To TcVnVector2_UINT;
  ipContainer : ITCvNContainer;
  hrPrev : HRESULT;
END_VAR
```

Append a single element of type TcVnVector2_UINT to a container.

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector2_UINT [153]</td>
<td>Single element to append to ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer [345]</td>
<td>Container to which the element will be appended</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]
**.api reference**

**Required License**
TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.1.21 F_VN_AppendToContainer_TcVnVector2_USINT

Appends a single element of type `TcVnVector2_USINT` to a container.

#### Syntax

**Definition:**

```c
FUNCTION F_VN_AppendToContainer_TcVnVector2_USINT : HRESULT
VAR_INPUT
    aElement : Reference To TcVnVector2_USINT;
    ipContainer : ITCvnContainer;
    hrPrev : HRESULT;
END_VAR
```

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector2_USINT</td>
<td>Single element to append to ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvnContainer [153]</td>
<td>Container to which the element will be appended</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

#### Return value

`HRESULT [135]`

**Required License**
TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
### 6.4.4.1.22 F_VN_AppendDateToContainer_TcVnVector3_INT

Append a single element of type TcVnVector3_INT to a container.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_AppendDateToContainer_TcVnVector3_INT : HRESULT
VAR_INPUT
    aElement   : Reference To TcVnVector3_INT;
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector3_INT</td>
<td>Single element to append to ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container to which the element will be appended</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.1.23 F_VN_AppendDateToContainer_TcVnVector3_REAL

Append a single element of type TcVnVector3_REAL to a container.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_AppendDateToContainer_TcVnVector3_REAL : HRESULT
VAR_INPUT
    aElement   : Reference To TcVnVector3_REAL;
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```
F_VN_AppendToContainer_TcVnVector3_SINT

Append a single element of type TcVnVector3_SINT to a container.

Syntax

Definition:

FUNCTION F_VN_AppendToContainer_TcVnVector3_SINT : HRESULT
VAR_INPUT
    aElement : Reference To TcVnVector3_SINT;
    ipContainer : ITCvNContainer;
    hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector3_SINT</td>
<td>Single element to append to ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer</td>
<td>Container to which the element will be appended</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.1.25  F_VN_AppendToContainer_TcVnVector3_UINT

Append a single element of type TcVnVector3_UINT to a container.

Syntax

Definition:

FUNCTION F_VN_AppendToContainer_TcVnVector3_UINT : HRESULT
VAR_INPUT
  aElement : Reference To TcVnVector3_UINT;
  ipContainer : ITCvnContainer;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector3_UINT [153]</td>
<td>Single element to append to ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvnContainer [345]</td>
<td>Container to which the element will be appended</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.4.1.26  F_VN_AppendToContainer_TcVnVector3_USINT

Append a single element of type TcVnVector3_USINT to a container.

Syntax

Definition:

FUNCTION F_VN_AppendToContainer_TcVnVector3_USINT : HRESULT
VAR_INPUT
   aElement : Reference To TcVnVector3_USINT;
   ipContainer : ITcVnContainer;
   hrPrev : HRESULT;
END_VAR

Inputs

Name | Type | Description
--- | --- | ---
aElement | Reference To TcVnVector3_USINT | Single element to append to ipContainer
ipContainer | ITcVnContainer | Container to which the element will be appended
hrPrev | HRESULT | HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.1.27  F_VN_AppendToContainer_TcVnVector4_DINT

Append a single element of type TcVnVector4_DINT to a container.

Syntax

Definition:

FUNCTION F_VN_AppendToContainer_TcVnVector4_DINT : HRESULT
VAR_INPUT
   aElement : Reference To TcVnVector4_DINT;
   ipContainer : ITcVnContainer;
   hrPrev : HRESULT;
END_VAR
Append a single element of type TcVnVector4_INT to a container.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_AppendToContainer_TcVnVector4_INT : HRESULT
VAR_INPUT
    aElement : Reference To TcVnVector4_INT;
    ipContainer : ITcVnContainer;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector4_INT</td>
<td>Single element to append to ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container to which the element will be appended</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g.</td>
<td>Tc3_Vision</td>
</tr>
<tr>
<td></td>
<td>Intel 4-core Atom CPU</td>
<td></td>
</tr>
</tbody>
</table>

### 6.4.4.1.29  F_VN_AppendToContainer_TcVnVector4_LREAL

Append a single element of type TcVnVector4_LREAL to a container.

**Syntax**

**Definition:**

FUNCTION F_VN_AppendToContainer_TcVnVector4_LREAL : HRESULT

VAR_INPUT

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Single element to append to ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvContainer [345]</td>
<td>Container to which the element will be appended</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g.</td>
<td>Tc3_Vision</td>
</tr>
<tr>
<td></td>
<td>Intel 4-core Atom CPU</td>
<td></td>
</tr>
</tbody>
</table>
6.4.4.1.30  F_VN_AppendToContainer_TcVnVector4_SINT

Append a single element of type TcVnVector4_SINT to a container.

Syntax

Definition:
FUNCTION F_VN_AppendToContainer_TcVnVector4_SINT : HRESULT
VAR_INPUT
  aElement    : Reference To TcVnVector4_SINT;
  ipContainer : ITcVnContainer;
  hrPrev      : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector4_SINT [153]</td>
<td>Single element to append to ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container to which the element will be appended</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.1.31  F_VN_AppendToContainer_TcVnVector4_UINT

Append a single element of type TcVnVector4_UINT to a container.

Syntax

Definition:
FUNCTION F_VN_AppendToContainer_TcVnVector4_UINT : HRESULT
VAR_INPUT
  aElement    : Reference To TcVnVector4_UINT;
  ipContainer : ITcVnContainer;
  hrPrev      : HRESULT;
END_VAR
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector4_USINT</td>
<td>Single element to append to ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container to which the element will be appended</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**F_VN_AppendToContainer_TcVnVector4_USINT**

Append a single element of type TcVnVector4_USINT to a container.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_AppendToContainer_TcVnVector4_USINT : HRESULT
VAR_INPUT
    aElement : Reference To TcVnVector4_USINT;
    ipContainer : ITcVnContainer;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector4_USINT</td>
<td>Single element to append to ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container to which the element will be appended</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]
6.4.4.1.33  F_VN_AppendToContainer_UDINT

Append a single element of type UDINT to a container.

Syntax

Definition:

```FUNCTION F_VN_AppendToContainer_UDINT : HRESULT
VAR_INPUT
    nElement : UDINT;
    ipContainer : ITcVnContainer;
    hrPrev : HRESULT;
END_VAR
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nElement</td>
<td>UDINT</td>
<td>Single element to append to ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container to which the element will be appended</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.4.1.34 F_VN_AppendToContainer_UINT

Append a single element of type UINT to a container.

Syntax

Definition:
FUNCTION F_VN_AppendToContainer_UINT : HRESULT
VAR_INPUT
  nElement : UINT;
  ipContainer : ITcVnContainer;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nElement</td>
<td>UINT</td>
<td>Single element to append to ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container to which the element will be appended</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.1.35 F_VN_AppendToContainer_ULINT

Append a single element of type ULINT to a container.

Syntax

Definition:
FUNCTION F_VN_AppendToContainer_ULINT : HRESULT
VAR_INPUT
  nElement : ULINT;
  ipContainer : ITcVnContainer;
  hrPrev : HRESULT;
END_VAR
Append a single element of type USINT to a container.

Syntax

Definition:

FUNCTION F_VN_AppendToContainer_USINT : HRESULT
VAR_INPUT
   nElement : USINT;
   ipContainer : ITcVnContainer;
   hrPrev : HRESULT;
END_VAR
Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.2 F_VN_FillContainer

6.4.4.2.1 F_VN_FillContainer_DINT

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>F_VN_FillContainer_DINT</td>
<td>ipContainer</td>
<td>ITcVnContainer</td>
</tr>
<tr>
<td></td>
<td>nValue</td>
<td>DINT</td>
</tr>
<tr>
<td></td>
<td>hrPrev</td>
<td>HRESULT</td>
</tr>
</tbody>
</table>

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

FUNCTION F_VN_FillContainer_DINT : HRESULT
VAR_INPUT
   ipContainer : ITcVnContainer;
   nValue : DINT;
   hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>nValue</td>
<td>DINT</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.2.2  **F_VN_FillContainer_INT**

```plaintext
F_VN_FillContainer_INT

---

ipContainer : ITcVnContainer
nValue : INT
hrPrev : HRESULT
```

Fill the container with the specified value. Only possible for containers with basic elements.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_FillContainer_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nValue : INT;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>nValue</td>
<td>INT</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.2.3  **F_VN_FillContainer_LREAL**

```plaintext
F_VN_FillContainer_LREAL

---

ipContainer : ITcVnContainer
fValue : LREAL
hrPrev : HRESULT
```

Fill the container with the specified value. Only possible for containers with basic elements.
Syntax

**Definition:**

```plaintext
FUNCTION F_VN_FillContainer_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    fValue      : LREAL;
    hrPrev      : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [135]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>fValue</td>
<td>LREAL</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.2.4 F_VN_FillContainer_REAL

F_VN_FillContainer_REAL

- ipContainer: ITcVnContainer
- fValue: REAL
- hrPrev: HRESULT

Fill the container with the specified value. Only possible for containers with basic elements.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_FillContainer_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    fValue      : REAL;
    hrPrev      : HRESULT;
END_VAR
```
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>fValue</td>
<td>REAL</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.2.5  F_VN_FillContainer_SINT

F_VN_FillContainer_SINT

F_VN_FillContainer_SINT

<table>
<thead>
<tr>
<th>ipContainer</th>
<th>ITcVnContainer</th>
<th>HRESULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>nValue</td>
<td>SINT</td>
<td></td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td></td>
</tr>
</tbody>
</table>

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

FUNCTION F_VN_FillContainer_SINT : HRESULT

VAR_INPUT

   ipContainer : ITcVnContainer;
   nValue : SINT;
   hrPrev : HRESULT;

END_VAR

Return value

HRESULT [135]
### Required License
TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.2.6 **F_VN_FillContainer_TcVnDMatch**

F_VN_FillContainer_TcVnDMatch

- **ipContainer**: ITCvNContainer
- **stValue**: Reference To TcVnDMatch
- **hrPrev**: HRESULT

Fill the container with the specified value. Only possible for containers with basic elements.

### Syntax

**Definition:**

```c
FUNCTION F_VN_FillContainer_TcVnDMatch : HRESULT
VAR_INPUT
    ipContainer : ITCvNContainer;
    stValue : Reference To TcVnDMatch;
    hrPrev : HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>stValue</td>
<td>Reference To TcVnDMatch</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

**HRESULT**

### Required License
TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.4.2.7  F_VN_FillContainer_TcVnKeyPoint

**Syntax**

**Definition:**

```c
FUNCTION F_VN_FillContainer_TcVnKeyPoint : HRESULT
VAR_INPUT
ipContainer : ITcVnContainer;
stValue      : Reference To TcVnKeyPoint;
hrPrev       : HRESULT;
END_VAR
```  

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>stValue</td>
<td>Reference To TcVnKeyPoint</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

**HRESULT [135]**

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.2.8  F_VN_FillContainer_TcVnPoint2_DINT

**Syntax**

**Definition:**

```c
FUNCTION F_VN_FillContainer_TcVnPoint2_DINT : HRESULT
VAR_INPUT
ipContainer : ITcVnContainer;
alValue     : Reference To TcVnPoint2_DINT;
hrPrev       : HRESULT;
END_VAR
```  

Fill the container with the specified value. Only possible for containers with basic elements.
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [1345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnPoint2_DINT [151]</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.4.2.9 F_VN_FillContainer_TcVnPoint2_LREAL**

Fill the container with the specified value. Only possible for containers with basic elements.

**Syntax**

**Definition:**

```
FUNCTION F_VN_FillContainer_TcVnPoint2_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aValue     : Reference To TcVnPoint2_LREAL;
    hrPrev     : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [1345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnPoint2_LREAL [151]</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.2.10  F_VN_FillContainer_TcVnPoint2_REAL

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```c
FUNCTION F_VN_FillContainer_TcVnPoint2_REAL : HRESULT
VAR_INPUT
    ipContainer : ITCVnContainer;
    aValue     : Reference To TcVnPoint2_REAL;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCVnContainer [345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnPoint2_REAL [151]</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.4.2.11 F_VN_FillContainer_TcVnPoint3_LREAL

Fill the container with the specified value. Only possible for containers with basic elements.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_FillContainer_TcVnPoint3_LREAL : HRESULT
VAR_INPUT
  ipContainer : ITcVnContainer;
  aValue      : Reference To TcVnPoint3_LREAL;
  hrPrev      : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnPoint3_LREAL</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.2.12 F_VN_FillContainer_TcVnPoint3_REAL

Fill the container with the specified value. Only possible for containers with basic elements.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_FillContainer_TcVnPoint3_REAL : HRESULT
VAR_INPUT
  ipContainer : ITcVnContainer;
  aValue      : Reference To TcVnPoint3_REAL;
  hrPrev      : HRESULT;
END_VAR
```
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [ ▶ 345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnPoint3_REAL [ ▶ 151]</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [ ▶ 135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [ ▶ 135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### Syntax

**Definition:**

FUNCTION F_VN_FillContainer_TcVnRectangle_DINT : HRESULT
VAR_INPUT
  ipContainer : ITcVnContainer;
  stValue : Reference To TcVnRectangle_DINT;
  hrPrev : HRESULT;
END_VAR

Fill the container with the specified value. Only possible for containers with basic elements.

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [ ▶ 345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>stValue</td>
<td>Reference To TcVnRectangle_DINT [ ▶ 221]</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [ ▶ 135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.2.14 F_VN_FillContainer_TcVnVector2_DINT

**F_VN_FillContainer_TcVnVector2_DINT**

**Syntax**

**Definition:**

FUNCTION F_VN_FillContainer_TcVnVector2_DINT : HRESULT
VAR_INPUT
  ipContainer : ITCvnContainer;
  aValue     : Reference To TcVnVector2_DINT;
  hrPrev     : HRESULT;
END_VAR

Fill the container with the specified value. Only possible for containers with basic elements.

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvnContainer [345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector2_DINT [153]</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (if SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

---

548   Version: 1.3   TF7000 - TF7300
### 6.4.4.2.15 F_VN_FillContainer_TcVnVector2_INT

**F_VN_FillContainer_TcVnVector2_INT**

*ipContainer* : *ITcVnContainer*

*aValue* : *Reference To TcVnVector2_INT*

*hrPrev* : *HRESULT*

Fill the container with the specified value. Only possible for containers with basic elements.

**Syntax**

**Definition:**

```
FUNCTION F_VN_FillContainer_TcVnVector2_INT : HRESULT
VAR_INPUT
   ipContainer : ITcVnContainer;
   aValue      : Reference To TcVnVector2_INT;
   hrPrev      : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td><em>ITcVnContainer</em> [345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td><em>Reference To TcVnVector2_INT</em> [153]</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td><em>HRESULT</em> [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

*HRESULT* [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.2.16 F_VN_FillContainer_TcVnVector2_REAL

**F_VN_FillContainer_TcVnVector2_REAL**

*ipContainer* : *ITcVnContainer*

*aValue* : *Reference To TcVnVector2_REAL*

*hrPrev* : *HRESULT*

Fill the container with the specified value. Only possible for containers with basic elements.

**Syntax**

**Definition:**

```
FUNCTION F_VN_FillContainer_TcVnVector2_REAL : HRESULT
VAR_INPUT
   ipContainer : ITcVnContainer;
```

---

TF7000 - TF7300  
Version: 1.3  
549
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector2_SINT</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.4.2.17 F_VN_FillContainer_TcVnVector2_SINT**

Fill the container with the specified value. Only possible for containers with basic elements.

**Syntax**

Definition:

```
FUNCTION F_VN_FillContainer_TcVnVector2_SINT : HRESULT
VAR_INPUT
  ipContainer : ITcVnContainer;
  aValue      : Reference To TcVnVector2_SINT;
  hrPrev      : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector2_SINT</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.2.18  F_VN_FillContainer_TcVnVector2_UINT

```plaintext
FUNCTION F_VN_FillContainer_TcVnVector2_UINT : HRESULT
VAR_INPUT
  ipContainer : ITCvnContainer;
  aValue      : Reference To TcVnVector2_UINT;
  hrPrev      : HRESULT;
END_VAR
```

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

FUNCTION F_VN_FillContainer_TcVnVector2_UINT : HRESULT
VAR_INPUT
  ipContainer : ITCvnContainer;
  aValue      : Reference To TcVnVector2_UINT;
  hrPrev      : HRESULT;
END_VAR

Inputs

| Name     | Type                                      | Description                                                         |
|----------|-------------------------------------------|                                                                    |
| ipContainer | ITCvnContainer [345]                   | Container with basic elements                                        |
| aValue      | Reference To TcVnVector2_UINT [153]     | Value to set the container elements                                 |
| hrPrev      | HRESULT [135]                           | HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.) |

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
### F_VN_FillContainer_TcVnVector2_USINT

**Definition:**

```plaintext
FUNCTION F_VN_FillContainer_TcVnVector2_USINT : HRESULT
VAR_INPUT
    ipContainer : ITCvnContainer;
    aValue : Reference To TcVnVector2_USINT;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvnContainer [345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector2_USINT [153]</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

### F_VN_FillContainer_TcVnVector3_INT

**Definition:**

```plaintext
FUNCTION F_VN_FillContainer_TcVnVector3_INT : HRESULT
VAR_INPUT
    ipContainer : ITCvnContainer;
    aValue : Reference To TcVnVector3_INT [153];
    hrPrev : HRESULT;
END_VAR
```

Fill the container with the specified value. Only possible for containers with basic elements.
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td></td>
<td>TcVnVector3_INT [153]</td>
<td></td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.2.21 F_VN_FillContainer_TcVnVector3_REAL

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

FUNCTION F_VN_FillContainer_TcVnVector3_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aValue     : Reference To TcVnVector3_REAL;
    hrPrev      : HRESULT;
END_VAR
Return value

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.2.22 **F_VN_FillContainer_TcVnVector3_SINT**

Fill the container with the specified value. Only possible for containers with basic elements.

**Syntax**

**Definition:**

FUNCTION F_VN_FillContainer_TcVnVector3_SINT : HRESULT
VAR_INPUT
  ipContainer : ITCvVnContainer;
  aValue     : Reference To TcVnVector3_SINT;
  hrPrev      : HRESULT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvVnContainer [345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector3_SINT [153]</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.4.2.23 F_VN_FillContainer_TcVnVector3_UINT

**F_VN_FillContainer_TcVnVector3_UINT**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector3_UINT</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
</tr>
</tbody>
</table>

Fill the container with the specified value. Only possible for containers with basic elements.

**Syntax**

**Definition:**

`FUNCTION F_VN_FillContainer_TcVnVector3_UINT : HRESULT`

```c
VAR_INPUT
    ipContainer : ITcVnContainer;
    aValue     : Reference To TcVnVector3_UINT;
    hrPrev     : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector3_UINT [153]</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.2.24 F_VN_FillContainer_TcVnVector3_USINT

**F_VN_FillContainer_TcVnVector3_USINT**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector3_USINT</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
</tr>
</tbody>
</table>

Fill the container with the specified value. Only possible for containers with basic elements.

**Syntax**

**Definition:**

`FUNCTION F_VN_FillContainer_TcVnVector3_USINT : HRESULT`

```c
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR`
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [1345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector3_USINT [153]</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.2.25 F_VN_FillContainer_TcVnVector4_DINT

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```plaintext
FUNCTION F_VN_FillContainer_TcVnVector4_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aValue : Reference To TcVnVector4_DINT;
    hrPrev : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [1345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector4_DINT [153]</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.2.26 F_VN_FillContainer_TcVnVector4_INT

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

FUNCTION F_VN_FillContainer_TcVnVector4_INT : HRESULT
VAR_INPUT
  ipContainer : ITCvContainer;
  aValue : Reference To TcVnVector4_INT;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvContainer</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector4_INT</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.4.2.27  F_VN_FillContainer_TcVnVector4_LREAL

**Syntax**

**Definition:**

```c
FUNCTION F_VN_FillContainer_TcVnVector4_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITCvNContainer;
    aValue : Reference To TcVnVector4_LREAL;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.2.28  F_VN_FillContainer_TcVnVector4_SINT

**Syntax**

**Definition:**

```c
FUNCTION F_VN_FillContainer_TcVnVector4_SINT : HRESULT
VAR_INPUT
    ipContainer : ITCvNContainer;
END_VAR
```

Fill the container with the specified value. Only possible for containers with basic elements.
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector4_UINT [153]</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g.</td>
<td>Tc3_Vision</td>
</tr>
<tr>
<td></td>
<td>Intel 4-core Atom CPU</td>
<td></td>
</tr>
</tbody>
</table>

6.4.4.2.29 F_VN_FillContainer_TcVnVector4_UINT

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

FUNCTION F_VN_FillContainer_TcVnVector4_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aValue     : Reference To TcVnVector4_UINT;
    hrPrev      : HRESULT;
END_VAR
Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.2.30 F_VN_FillContainer_TcVnVector4_USINT

```
F_VN_FillContainer_TcVnVector4_USINT
ipContainer : ITCvVnContainer
aValue : Reference To TcVnVector4_USINT
hrPrev : HRESULT
```

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_TcVnVector4_USINT : HRESULT
VAR_INPUT
    ipContainer : ITCvVnContainer;
    aValue : Reference To TcVnVector4_USINT;
    hrPrev : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvVnContainer [345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector4_USINT [153]</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.4.2.31  F_VN_FillContainer_UDINT

**F_VN_FillContainer_UDINT**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>nValue</td>
<td>UDINT</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Fill the container with the specified value. Only possible for containers with basic elements.

**Syntax**

Definition:

```plaintext
FUNCTION F_VN_FillContainer_UDINT : HRESULT
VAR_INPUT
    ipContainer     : ITcVnContainer;
    nValue          : UDINT;
    hrPrev          : HRESULT;
END_VAR
```

**Inputs**

- **ipContainer**: ITcVnContainer
  - **Type**: Container with basic elements
- **nValue**: UDINT
  - **Type**: Value to set the container elements
- **hrPrev**: HRESULT
  - **Type**: HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.2.32  F_VN_FillContainer_UINT

**F_VN_FillContainer_UINT**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td></td>
</tr>
<tr>
<td>nValue</td>
<td>UINT</td>
<td></td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td></td>
</tr>
</tbody>
</table>

Fill the container with the specified value. Only possible for containers with basic elements.

**Syntax**

Definition:

```plaintext
FUNCTION F_VN_FillContainer_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
```

**Inputs**

- **ipContainer**: ITcVnContainer
  - **Type**: Container with basic elements
- **nValue**: UINT
  - **Type**: Value to set the container elements
- **hrPrev**: HRESULT
  - **Type**: HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>nValue</td>
<td>ULINT</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### Syntax

**Definition:**

```c
FUNCTION F_VN_FillContainer_ULINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nValue      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Fill the container with the specified value. Only possible for containers with basic elements.

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>nValue</td>
<td>ULINT</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.2.34 F_VN_FillContainer_USINT

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:
FUNCTION F_VN_FillContainer_USINT : HRESULT
VAR_INPUT
  ipContainer : ITcVnContainer;
  nValue : USINT;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>nValue</td>
<td>USINT</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.4.3  F_VN_FillContainerExp

This chapter contains expert functions that can be used to fill a container with a specific value, even only partially, according to the data type. This function is available for containers with basic elements.

6.4.4.3.1  F_VN_FillContainerExp_DINT

<table>
<thead>
<tr>
<th>F_VN_FillContainerExp_DINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
</tr>
<tr>
<td>nValue</td>
</tr>
<tr>
<td>nStartIdx</td>
</tr>
<tr>
<td>nCount</td>
</tr>
<tr>
<td>hrPrev</td>
</tr>
</tbody>
</table>

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_FillContainerExp_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nValue : DINT;
    nStartIdx : ULINT;
    nCount : ULINT;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>nValue</td>
<td>DINT</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
### F_VN_FillContainerExp_INT

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_FillContainerExp_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nValue      : INT;
    nStartIdx   : ULONG;
    nCount      : ULONG;
    hrPrev      : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>nValue</td>
<td>INT</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULONG</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULONG</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### F_VN_FillContainerExp_LREAL
## Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

### Syntax

**Definition:**

```c
FUNCTION F_VN_FillContainerExp_LREAL : HRESULT
VAR_INPUT
   ipContainer : ITcVnContainer;
   fValue      : LREAL;
   nStartIdx   : ULINT;
   nCount      : ULINT;
   hrPrev      : HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>fValue</td>
<td>REAL</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

**HRESULT [135]**

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.3.4 F_VN_FillContainerExp_REAL

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

**Syntax**

**Definition:**

```c
FUNCTION F_VN_FillContainerExp_REAL : HRESULT
VAR_INPUT
   ipContainer : ITcVnContainer;
   fValue      : REAL;
END_VAR
```
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>fValue</td>
<td>REAL</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

**HRESULT [135]**

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.3.5 F_VN_FillContainerExp_SINT

```c
FUNCTION F_VN_FillContainerExp_SINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nValue      : SINT;
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

### Syntax

**Definition:**

```c
FUNCTION F_VN_FillContainerExp_SINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nValue      : SINT;
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer ![345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>nValue</td>
<td>SINT</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT ![135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT ![135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.3.6 F_VN_FillContainerExp_TcVnDMatch

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

**Syntax**

**Definition:**

```
FUNCTION F_VN_FillContainerExp_TcVnDMatch : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    stValue      : Reference To TcVnDMatch;
    nStartIdx    : ULINT;
    nCount       : ULINT;
    hrPrev       : HRESULT;
END_VAR
```
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [IP_345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>stValue</td>
<td>Reference To TcVnDMatch [IP_207]</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [IP_135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [IP_135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.3.7 F_VN_FillContainerExp_TcVnKeyPoint

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

FUNCTION F_VN_FillContainerExp_TcVnKeyPoint : HRESULT
VAR_INPUT
   ipContainer : ITcVnContainer;
   stValue : Reference To TcVnKeyPoint;
   nStartIdx : ULINT;
   nCount : ULINT;
   hrPrev : HRESULT;
END_VAR
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>stValue</td>
<td>Reference To TcVnKeyPoint [208]</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

#### 6.4.4.3.8 F_VN_FillContainerExp_TcVnPoint2_DINT

```plaintext
F_VN_FillContainerExp_TcVnPoint2_DINT
```

- **ipContainer** : ITcVnContainer
- **aValue** : Reference To TcVnPoint2_DINT
- **nStartIdx** : ULINT
- **nCount** : ULINT
- **hrPrev** : HRESULT

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_FillContainerExp_TcVnPoint2_DINT : HRESULT
VAR_INPUT
   ipContainer : ITcVnContainer;
   aValue      : Reference To TcVnPoint2_DINT;
   nStartIdx   : ULINT;
   nCount      : ULINT;
   hrPrev      : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer [345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnPoint2_DINT [151]</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>UINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>UINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.3.9  F_VN_FillContainerExp_TcVnPoint2_LREAL

```c
FUNCTION F_VN_FillContainerExp_TcVnPoint2_LREAL : HRESULT
VAR_INPUT
  ipContainer : ITCvNContainer;
  aValue : Reference To TcVnPoint2_LREAL;
  nStartIdx : ULINT;
  nCount : ULINT;
  hrPrev : HRESULT;
END_VAR
```

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

### Syntax

**Definition:**

FUNCTION F_VN_FillContainerExp_TcVnPoint2_LREAL : HRESULT
VAR_INPUT
  ipContainer : ITCvNContainer;
  aValue : Reference To TcVnPoint2_LREAL;
  nStartIdx : ULINT;
  nCount : ULINT;
  hrPrev : HRESULT;
END_VAR
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer[134]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnPoint2_REAL[151]</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT[135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.3.10 F_VN_FillContainerExp_TcVnPoint2_REAL

```c
FUNCTION F_VN_FillContainerExp_TcVnPoint2_REAL : HRESULT
VAR_INPUT
   ipContainer : ITcVnContainer;
   aValue : Reference To TcVnPoint2_REAL;
   nStartIdx : ULINT;
   nCount : ULINT;
   hrPrev : HRESULT;
END_VAR
```

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

FUNCTION F_VN_FillContainerExp_TcVnPoint2_REAL : HRESULT
VAR_INPUT
   ipContainer : ITcVnContainer;
   aValue : Reference To TcVnPoint2_REAL;
   nStartIdx : ULINT;
   nCount : ULINT;
   hrPrev : HRESULT;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer [345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnPoint2_REAL [151]</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.3.11 F_VN_FillContainerExp_TcVnPoint3_LREAL

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

#### Syntax

**Definition:**

```c
FUNCTION F_VN_FillContainerExp_TcVnPoint3_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITCvNContainer;
    aValue     : Reference To TcVnPoint3_LREAL;
    nStartIdx  : ULINT;
    nCount     : ULINT;
    hrPrev     : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvnContainer <img src="https://example.com" alt="345" /></td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnPoint3_REAL <img src="https://example.com" alt="151" /></td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT <img src="https://example.com" alt="135" /></td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT ![135](https://example.com)

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.3.12 F_VN_FillContainerExp_TcVnPoint3_REAL

<table>
<thead>
<tr>
<th>ipContainer</th>
<th>ITCvnContainer</th>
</tr>
</thead>
<tbody>
<tr>
<td>aValue</td>
<td>Reference To TcVnPoint3_REAL</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT <img src="https://example.com" alt="135" /></td>
</tr>
</tbody>
</table>

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

### Syntax

**Definition:**

```c
FUNCTION F_VN_FillContainerExp_TcVnPoint3_REAL : HRESULT
VAR_INPUT
  ipContainer : ITCvnContainer;
  aValue : Reference To TcVnPoint3_REAL;
  nStartIdx : ULINT;
  nCount : ULINT;
  hrPrev : HRESULT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnPoint3_REAL</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.3.13  **F_VN_FillContainerExp_TcVnRectangle_DINT**

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

**Syntax**

**Definition:**

```c
FUNCTION F_VN_FillContainerExp_TcVnRectangle_DINT : HRESULT
VAR_INPUT
   ipContainer : ITcVnContainer;
   stValue     : Reference To TcVnRectangle_DINT;
   nStartIdx   : ULINT;
   nCount      : ULINT;
   hrPrev      : HRESULT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>stValue</td>
<td>Reference To</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td></td>
<td>TcVnRectangle_DINT</td>
<td>CADES</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.3.14 **F_VN_FillContainerExp_TcVnVector2_DINT**

```plaintext
ipContainer : ITcVnContainer;
newValue : Reference To TcVnVector2_DINT;
nStartIdx : ULINT;
nCount : ULINT;
hrPrev : HRESULT;
```

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_FillContainerExp_TcVnVector2_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    newValue : Reference To TcVnVector2_DINT;
    nStartIdx : ULINT;
    nCount : ULINT;
    hrPrev : HRESULT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector2_DINT</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.3.15  **F_VN_FillContainerExp_TcVnVector2_INT**

```
FUNCTION F_VN_FillContainerExp_TcVnVector2_INT : HRESULT
VAR_INPUT
   ipContainer : ITcVnContainer;
   aValue : Reference To TcVnVector2_INT;
   nStartIdx : ULINT;
   nCount : ULINT;
   hrPrev : HRESULT;
END_VAR
```

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

**Syntax**

**Definition:**

FUNCTION F_VN_FillContainerExp_TcVnVector2_INT : HRESULT
VAR_INPUT
   ipContainer : ITcVnContainer;
   aValue : Reference To TcVnVector2_INT;
   nStartIdx : ULINT;
   nCount : ULINT;
   hrPrev : HRESULT;
END_VAR
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [134]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector2_INT [153]</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.3.16  F_VN_FillContainerExp_TcVnVector2_REAL

<table>
<thead>
<tr>
<th>ipContainer</th>
<th>ITcVnContainer</th>
<th>aValue</th>
<th>Reference To TcVnVector2_REAL</th>
<th>nStartIdx</th>
<th>ULINT</th>
<th>nCount</th>
<th>ULINT</th>
<th>hrPrev</th>
<th>HRESULT</th>
</tr>
</thead>
</table>

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

**Syntax**

**Definition:**

FUNCTION F_VN_FillContainerExp_TcVnVector2_REAL : HRESULT

VAR_INPUT
  ipContainer : ITcVnContainer;
  aValue : Reference To TcVnVector2_REAL;
  nStartIdx : ULINT;
  nCount : ULINT;
  hrPrev : HRESULT;
END_VAR
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [134]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector2_REAL [153]</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULLINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULLINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.4.3.17 F_VN_FillContainerExp_TcVnVector2_SINT**

```plaintext
FUNCTION F_VN_FillContainerExp_TcVnVector2_SINT : HRESULT
VAR_INPUT
  ipContainer : ITcVnContainer;
  aValue     : Reference To TcVnVector2_SINT;
  nStartIdx  : ULLINT;
  nCount     : ULLINT;
  hrPrev     : HRESULT;
END_VAR
```

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

**Syntax**

**Definition:**

FUNCTION F_VN_FillContainerExp_TcVnVector2_SINT : HRESULT
VAR_INPUT
  ipContainer : ITcVnContainer;
  aValue     : Reference To TcVnVector2_SINT;
  nStartIdx  : ULLINT;
  nCount     : ULLINT;
  hrPrev     : HRESULT;
END_VAR
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector2_SINT [153]</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.3.18 F_VN_FillContainerExp_TcVnVector2_UINT

```
FUNCTION F_VN_FillContainerExp_TcVnVector2_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aValue      : Reference To TcVnVector2_UINT;
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

### Syntax

**Definition:**

FUNCTION F_VN_FillContainerExp_TcVnVector2_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aValue      : Reference To TcVnVector2_UINT;
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector2_UINT</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.3.19 F_VN_FillContainerExp_TcVnVector2_USINT

```
F_VN_FillContainerExp_TcVnVector2_USINT
```

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_TcVnVector2_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aValue : Reference To TcVnVector2_USINT;
    nStartIdx : ULINT;
    nCount : ULINT;
    hrPrev : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCVnContainer</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector2 USINT</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.4.3.20 F_VN_FillContainerExp_TcVnVector3_INT

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_FillContainerExp_TcVnVector3_INT : HRESULT
VAR_INPUT
    ipContainer : ITCVnContainer;
    aValue     : Reference To TcVnVector3_INT;
    nStartIdx  : ULINT;
    nCount     : ULINT;
    hrPrev     : HRESULT;
END_VAR
```
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer[1.345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector3_INT[1.153]</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[1.135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT[1.135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.3.21  F_VN_FillContainerExp_TcVnVector3_REAL

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

### Syntax

**Definition:**

```c
FUNCTION F_VN_FillContainerExp_TcVnVector3_REAL : HRESULT
VAR_INPUT
  ipContainer : ITcVnContainer;
  aValue : Reference To TcVnVector3_REAL;
  nStartIdx : ULINT;
  nCount : ULINT;
  hrPrev : HRESULT;
END_VAR
```
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer [345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector3_REAL [153]</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.3.22 F_VN_FillContainerExp_TcVnVector3_SINT

```
F_VN_FillContainerExp_TcVnVector3_SINT
```

ipContainer : ITCvNContainer;
aValue : Reference To TcVnVector3_SINT;
nStartIdx : ULINT;
nCount : ULINT;
hrPrev : HRESULT;

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

**Definition:**

```c
FUNCTION F_VN_FillContainerExp_TcVnVector3_SINT : HRESULT
VAR_INPUT
  ipContainer : ITCvNContainer;
  aValue : Reference To TcVnVector3_SINT;
  nStartIdx : ULINT;
  nCount : ULINT;
  hrPrev : HRESULT;
END_VAR
```
**API reference**

TF7000 - TF7300

Version: 1.3

---

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvVContainer [_345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector3_SINT [_153]</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [\_135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

---

### 6.4.4.3.23 F_VN_FillContainerExp_TcVnVector3_UINT

```plaintext
F_VN_FillContainerExp_TcVnVector3_UINT
```

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_FillContainerExp_TcVnVector3_UINT : HRESULT
VAR_INPUT
    ipContainer : ITCvVContainer;
    aValue     : Reference To TcVnVector3_UINT;
    nStartIdx  : ULINT;
    nCount     : ULINT;
    hrPrev     : HRESULT;
END_VAR
```

---

TF7000 - TF7300

Version: 1.3

---

585
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer [345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector3_UINT [153]</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.4.3.24 F_VN_FillContainerExp_TcVnVector3_USINT**

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

**Syntax**

**Definition:**

FUNCTION F_VN_FillContainerExp_TcVnVector3_USINT : HRESULT

VAR_INPUT

- ipContainer : ITCvNContainer;
- aValue : Reference To TcVnVector3_USINT;
- nStartIdx : ULINT;
- nCount : ULINT;
- hrPrev : HRESULT;

END_VAR
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector3 USINT [153]</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### Syntax

6.4.4.3.25 **F_VN_FillContainerExp_TcVnVector4_DINT**

```
FUNCTION F_VN_FillContainerExp_TcVnVector4_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aValue : Reference To TcVnVector4_DINT;
    nStartIdx : ULINT;
    nCount : ULINT;
    hrPrev : HRESULT;
END_VAR
```

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector4 DINT</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.3.26 F_VN_FillContainerExp_TcVnVector4_INT

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

#### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_FillContainerExp_TcVnVector4_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aValue : Reference To TcVnVector4_INT;
    nStartIdx : ULINT;
    nCount : ULINT;
    hrPrev : HRESULT;
END_VAR
```
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector4_INT</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

**HRESULT [135]**

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.3.27 F_VN_FillContainerExp_TcVnVector4_LREAL

**F_VN_FillContainerExp_TcVnVector4_LREAL**

- **ipContainer** ITcVnContainer
- **aValue** Reference To TcVnVector4_LREAL
- **nStartIdx** ULINT
- **nCount** ULINT
- **hrPrev** HRESULT

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

### Syntax

**Definition:**

```c
FUNCTION F_VN_FillContainerExp_TcVnVector4_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aValue : Reference To TcVnVector4_LREAL;
    nStartIdx : ULINT;
    nCount : ULINT;
    hrPrev : HRESULT;
END_VAR
```
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

#### 6.4.4.3.28 F_VN_FillContainerExp_TcVnVector4_SINT

```
ipContainer ITcVnContainer
aValue Reference To TcVnVector4_SINT
nStartIdx ULINT
nCount ULINT
hrPrev HRESULT
```

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

### Syntax

**Definition:**

FUNCTION F_VN_FillContainerExp_TcVnVector4_SINT : HRESULT

VAR_INPUT

- ipContainer : ITcVnContainer;
- aValue : Reference To TcVnVector4_SINT;
- nStartIdx : ULINT;
- nCount : ULINT;
- hrPrev : HRESULT;

END_VAR
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer[345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector4_UINT[153]</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT[135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.3.29 **F_VN_FillContainerExp_TcVnVector4_UINT**

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

**Syntax**

Definition:

FUNCTION F_VN_FillContainerExp_TcVnVector4_UINT : HRESULT

VAR_INPUT

  ipContainer : ITcVnContainer;
  aValue : Reference To TcVnVector4_UINT;
  nStartIdx : ULINT;
  nCount : ULINT;
  hrPrev : HRESULT;

END_VAR
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector4_UINT [153]</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.3.30 **F_VN_FillContainerExp_TcVnVector4_USINT**

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_FillContainerExp_TcVnVector4_USINT : HRESULT
VAR_INPUT
  ipContainer : ITcVnContainer;
  aValue     : Reference To TcVnVector4_USINT;
  nStartIdx  : ULINT;
  nCount     : ULINT;
  hrPrev     : HRESULT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector4 USINT [153]</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.3.31 **F_VN_FillContainerExp_UDINT**

```
FUNCTION F_VN_FillContainerExp_UDINT : HRESULT
VAR_INPUT
  ipContainer : ITcVnContainer;
  nValue      : UDINT;
  nStartIdx   : ULINT;
  nCount      : ULINT;
  hrPrev      : HRESULT;
END_VAR
```

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

**Syntax**

**Definition:**

```
FUNCTION F_VN_FillContainerExp_UDINT : HRESULT
VAR_INPUT
  ipContainer : ITcVnContainer;
  nValue      : UDINT;
  nStartIdx   : ULINT;
  nCount      : ULINT;
  hrPrev      : HRESULT;
END_VAR
```
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>nValue</td>
<td>UDINT</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.3.32 F_VN_FillContainerExp_UINT

**F_VN_FillContainerExp_UINT**

- ipContainer : ITcVnContainer
- nValue : UINT
- nStartIdx : ULINT
- nCount : ULINT
- hrPrev : HRESULT

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

### Syntax

**Definition:**

```c
FUNCTION F_VN_FillContainerExp_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nValue : UINT;
    nStartIdx : ULINT;
    nCount : ULINT;
    hrPrev : HRESULT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>nValue</td>
<td>UINT</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.4.3.33 F_VN_FillContainerExp_ULINT**

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

**Syntax**

**Definition:**

```c
FUNCTION F_VN_FillContainerExp_ULINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nValue      : ULINT;
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>nValue</td>
<td>USINT</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.4.3.34 F_VN_FillContainerExp_USINT**

```
F_VN_FillContainerExp_USINT

ipContainer : ITcVnContainer;
nValue : USINT;
nStartIdx : ULINT;
nCount : ULINT;
hrPrev : HRESULT;
```

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

**Syntax**

**Definition:**

FUNCTION F_VN_FillContainerExp_USINT : HRESULT

VAR_INPUT

- ipContainer : ITcVnContainer;
- nValue : USINT;
- nStartIdx : ULINT;
- nCount : ULINT;
- hrPrev : HRESULT;

END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>nValue</td>
<td>USINT</td>
<td>Value to set the container elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULINT</td>
<td>Start index</td>
</tr>
<tr>
<td>nCount</td>
<td>ULINT</td>
<td>Number of elements to set</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.4 F_VN_GetAt

The F_VN_GetAt functions should be used if individual elements in a container have to be accessed. If all elements in a container are to be processed in succession, it is much better to use an iterator [706].

### 6.4.4.4.1 F_VN_GetAt_DINT

```
F_VN_GetAt_DINT
```

Get the DINT element at the specified index of the container.

## Syntax

**Definition:**

```c
FUNCTION F_VN_GetAt_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nElement   : Reference To DINT;
    nIndex     : ULINT;
    hrPrev     : HRESULT;
END_VAR
```
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with DINT elements</td>
</tr>
<tr>
<td>nElement</td>
<td>Reference To DINT</td>
<td>Returns the element at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g.</td>
<td>Tc3_Vision</td>
</tr>
<tr>
<td></td>
<td>Intel 4-core Atom CPU</td>
<td></td>
</tr>
</tbody>
</table>

6.4.4.4.2 F_VN_GetAt_INT

F_VN_GetAt_INT

ipContainer: ITcVnContainer  
nElement: Reference To INT  
nIndex: ULINT  
hrPrev: HRESULT

Gets the INT element at the specified index of the container.

Syntax

Definition:

FUNCTION F_VN_GetAt_INT : HRESULT
VAR_INPUT
  ipContainer : ITcVnContainer;
  nElement   : Reference To INT;
  nIndex     : ULINT;
  hrPrev     : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with INT elements</td>
</tr>
<tr>
<td>nElement</td>
<td>Reference To INT</td>
<td>Returns the element at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
**Return value**

HRESULT [\ref{135}]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.4.4.3  F_VN_GetAt_ITcVnContainer**

```
FUNCTION F_VN_GetAt_ITcVnContainer : HRESULT
VAR_INPUT
    ipSrcContainer : ITcVnContainer;
    ipDestContainer : Reference To ITcVnContainer;
    nIndex : ULI NT;
    hrPrev : HRESULT;
END_VAR
```

Gets the ITcVnContainer element at the specified index of the source container. (Alternatively use interface method .GetAt.)

**Syntax**

Definition:

```
FUNCTION F_VN_GetAt_ITcVnContainer : HRESULT
VAR_INPUT
    ipSrcContainer : ITcVnContainer;
    ipDestContainer : Reference To ITcVnContainer;
    nIndex : ULI NT;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcContainer</td>
<td>ITcVnContainer [\ref{345}]</td>
<td>Container with ITcVnContainer elements</td>
</tr>
<tr>
<td>ipDestContainer</td>
<td>Reference To ITcVnContainer [\ref{345}]</td>
<td>Returns the container at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULI NT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [\ref{135}]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Required License**

TC3 Vision Base
F_VN_GetAt_LREAL

F_VN_GetAt_LREAL

ipContainer  ITcVnContainer
fElement    Reference To LREAL
nIndex      ULINT
hrPrev      HRESULT

gets the LREAL element at the specified index of the container.

Syntax

Definition:

FUNCTION F_VN_GetAt_LREAL : HRESULT
VAR_INPUT
  ipContainer : ITcVnContainer;
  fElement    : Reference To LREAL;
  nIndex      : ULINT;
  hrPrev      : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with LREAL elements</td>
</tr>
<tr>
<td>fElement</td>
<td>Reference To LREAL</td>
<td>Returns the element at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.4.4.5  F_VN_GetAt_REAL

F_VN_GetAt_REAL

ipContainer  ITcVnContainer  HRESULT  F_VN_GetAt_REAL
fElement  Reference To REAL
nIndex  ULINT
hrPrev  HRESULT

Gets the REAL element at the specified index of the container.

Syntax

Definition:

FUNCTION F_VN_GetAt_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    fElement : Reference To REAL;
    nIndex : ULINT;
    hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with REAL elements</td>
</tr>
<tr>
<td>fElement</td>
<td>Reference To REAL</td>
<td>Returns the element at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.4.6  F_VN_GetAt_SINT

F_VN_GetAt_SINT

ipContainer  ITcVnContainer  HRESULT  F_VN_GetAt_SINT
nElement  Reference To SINT
nIndex  ULINT
hrPrev  HRESULT

Gets the SINT element at the specified index of the container.
API reference

Syntax

Definition:

FUNCTION F_VN_GetAt_SINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nElement    : Reference To SINT;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with SINT elements</td>
</tr>
<tr>
<td>nElement</td>
<td>Reference To SINT</td>
<td>Returns the element at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.4.7  F_VN_GetAt_TcVnDMatch

gets the TcVnDMatch element at the specified index of the container.

Syntax

Definition:

FUNCTION F_VN_GetAt_TcVnDMatch : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    stElement   : Reference To TcVnDMatch;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with TcVnDMatch elements</td>
</tr>
<tr>
<td>stElement</td>
<td>Reference To TcVnDMatch [207]</td>
<td>Returns the element at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.4.8 F_VN_GetAt_TcVnKeyPoint

```c
FUNCTION F_VN_GetAt_TcVnKeyPoint : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    stElement : Reference To TcVnKeyPoint;
    nIndex : ULINT;
    hrPrev : HRESULT;
END_VAR
```

Gets the TcVnKeyPoint element at the specified index of the container.

## Syntax

### Definition:

```c
FUNCTION F_VN_GetAt_TcVnKeyPoint : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    stElement : Reference To TcVnKeyPoint;
    nIndex : ULINT;
    hrPrev : HRESULT;
END_VAR
```
Return value

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.4.9 F_VN_GetAt_TcVnPoint2_DINT

Gets the TcVnPoint2_DINT element at the specified index of the container.

Syntax

Definition:

FUNCTION F_VN_GetAt_TcVnPoint2_DINT : HRESULT
VAR_INPUT
  ipContainer : ITCvVnContainer;
  aElement   : Reference To TcVnPoint2_DINT;
  nIndex     : ULINT;
  hrPrev     : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvVnContainer</td>
<td>Container with TcVnPoint2_DINT elements</td>
</tr>
<tr>
<td>aElement</td>
<td>Reference To TcVnPoint2_DINT</td>
<td>Returns the element at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License
TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.4.10 F_VN_GetAt_TcVnPoint2_LREAL

<table>
<thead>
<tr>
<th><strong>F_VN_GetAt_TcVnPoint2_LREAL</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer: ITCvNContainer</td>
</tr>
<tr>
<td>aElement: Reference To TcVnPoint2_LREAL</td>
</tr>
<tr>
<td>nIndex: ULINT</td>
</tr>
<tr>
<td>hrPrev: HRESULT</td>
</tr>
</tbody>
</table>

Gets the TcVnPoint2_LREAL element at the specified index of the container.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_GetAt_TcVnPoint2_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITCvNContainer;
    aElement: Reference To TcVnPoint2_LREAL;
    nIndex : ULINT;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer</td>
<td>Container with TcVnPoint2_LREAL elements</td>
</tr>
<tr>
<td>aElement</td>
<td>Reference To TcVnPoint2_LREAL</td>
<td>Returns the element at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.4.11 F_VN_GetAt_TcVnPoint2_REAL

F_VN_GetAt_TcVnPoint2_REAL

- ipContainer : ITCvNContainer
- aElement : Reference To TcVnPoint2_REAL
- nIndex : ULINT
- hrPrev : HRESULT

Gets the TcVnPoint2_REAL element at the specified index of the container.

Syntax

Definition:

FUNCTION F_VN_GetAt_TcVnPoint2_REAL : HRESULT
VAR_INPUT
  ipContainer : ITCvNContainer;
  aElement : Reference To TcVnPoint2_REAL;
  nIndex : ULINT;
  hrPrev : HRESULT;
END_VAR

- Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer [345]</td>
<td>Container with TcVnPoint2_REAL elements</td>
</tr>
<tr>
<td>aElement</td>
<td>Reference To TcVnPoint2_REAL [151]</td>
<td>Returns the element at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

- Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x864) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.12 F_VN_GetAt_TcVnPoint3_LREAL

F_VN_GetAt_TcVnPoint3_LREAL

- ipContainer : ITCvNContainer
- aElement : Reference To TcVnPoint3_LREAL
- nIndex : ULINT
- hrPrev : HRESULT

Gets the TcVnPoint3_LREAL element at the specified index of the container.
### Syntax

**Definition:**

```c
FUNCTION F_VN_GetAt_TcVnPoint3_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aElement    : Reference To TcVnPoint3_REAL;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with TcVnPoint3_REAL elements</td>
</tr>
<tr>
<td>aElement</td>
<td>Reference To TcVnPoint3_REAL [151]</td>
<td>Returns the element at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

`HRESULT [135]`

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

#### 6.4.4.4.13 F_VN_GetAt_TcVnPoint3_REAL

```
F_VN_GetAt_TcVnPoint3_REAL
    ipContainer  ITcVnContainer
    aElement     Reference To TcVnPoint3_REAL
    nIndex       ULINT
    hrPrev       HRESULT
```

Gets the TcVnPoint3_REAL element at the specified index of the container.

### Syntax

**Definition:**

```c
FUNCTION F_VN_GetAt_TcVnPoint3_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aElement    : Reference To TcVnPoint3_REAL;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with TcVnRectangle_DINT elements</td>
</tr>
<tr>
<td>stElement</td>
<td>Reference To TcVnRectangle_DINT</td>
<td>Returns the element at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.4.4.14 F_VN_GetAt_TcVnRectangle_DINT**

F_VN_GetAt_TcVnRectangle_DINT

ipContainer : ITcVnContainer

stElement : Reference To TcVnRectangle_DINT

nIndex : ULINT

hrPrev : HRESULT

Gets the TcVnRectangle_DINT element at the specified index of the container.

**Syntax**

**Definition:**

FUNCTION F_VN_GetAt_TcVnRectangle_DINT : HRESULT

VAR_INPUT

ipContainer : ITcVnContainer;

stElement : Reference To TcVnRectangle_DINT;

nIndex : ULINT;

hrPrev : HRESULT;

END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with TcVnRectangle_DINT elements</td>
</tr>
<tr>
<td>stElement</td>
<td>Reference To TcVnRectangle_DINT</td>
<td>Returns the element at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.4.15  F_VN_GetAt_TcVnVector2_DINT

F_VN_GetAt_TcVnVector2_DINT

- ipContainer : ITCvNContainer
- aElement : Reference To TcVnVector2_DINT
- nIndex : ULINT
- hrPrev : HRESULT

Gets the TcVnVector2_DINT element at the specified index of the container.

Syntax

Definition:
FUNCTION F_VN_GetAt_TcVnVector2_DINT : HRESULT
VAR_INPUT
- ipContainer : ITCvNContainer;
- aElement : Reference To TcVnVector2_DINT;
- nIndex : ULINT;
- hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer [345]</td>
<td>Container with TcVnVector2_DINT elements</td>
</tr>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector2_DINT [153]</td>
<td>Returns the element at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License
TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.4.16  F_VN_GetAt_TcVnVector2_INT

F_VN_GetAt_TcVnVector2_INT

- **ipContainer** : ITcVnContainer
- **aElement** : Reference To TcVnVector2_INT
- **nIndex** : ULLINT
- **hrPrev** : HRESULT

Gets the TcVnVector2_INT element at the specified index of the container.

**Syntax**

**Definition:**

```pascal
FUNCTION F_VN_GetAt_TcVnVector2_INT : HRESULT
VAR_INPUT
  ipContainer : ITcVnContainer;
  aElement    : Reference To TcVnVector2_INT;
  nIndex      : ULLINT;
  hrPrev       : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with TcVnVector2_INT elements</td>
</tr>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector2_INT</td>
<td>Returns the element at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULLINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.4.4.17  F_VN_GetAt_TcVnVector2_REAL

### Syntax

**Definition:**

```
FUNCTION F_VN_GetAt_TcVnVector2_REAL : HRESULT
VAR_INPUT
    ipContainer : ITCvVnContainer;
    aElement : Reference To TcVnVector2_REAL;
    nIndex : ULINT;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvVnContainer [345]</td>
<td>Container with TcVnVector2_REAL elements</td>
</tr>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector2_REAL [153]</td>
<td>Returns the element at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g.</td>
<td>Tc3_Vision</td>
</tr>
<tr>
<td></td>
<td>Intel 4-core Atom CPU</td>
<td></td>
</tr>
</tbody>
</table>

6.4.4.4.18  F_VN_GetAt_TcVnVector2_SINT

### Syntax

**Definition:**

```
FUNCTION F_VN_GetAt_TcVnVector2_SINT : HRESULT
VAR_INPUT
    ipContainer : ITCvVnContainer;
    aElement : Reference To TcVnVector2_SINT;
    nIndex : ULINT;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvVnContainer</td>
<td>HRESULT F_VN_GetAt_TcVnVector2_SINT</td>
</tr>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector2_SINT</td>
<td>Gets the TcVnVector2_SINT element at the specified index of the container.</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td></td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td></td>
</tr>
</tbody>
</table>
## Syntax

**Definition:**

```
FUNCTION F_VN_GetAt_TcVnVector2_SINT : HRESULT
VAR_INPUT
  ipContainer : ITcVnContainer;
  aElement    : Reference To TcVnVector2_SINT;
  nIndex      : ULINT;
  hrPrev      : HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with TcVnVector2_SINT elements</td>
</tr>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector2_SINT [153]</td>
<td>Returns the element at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.4.19 F_VN_GetAt_TcVnVector2_UINT

```
FUNCTION F_VN_GetAt_TcVnVector2_UINT : HRESULT
VAR_INPUT
  ipContainer : ITcVnContainer;
  aElement    : Reference To TcVnVector2_UINT;
  nIndex      : ULINT;
  hrPrev      : HRESULT;
END_VAR
```

Gets the TcVnVector2_UINT element at the specified index of the container.

**Syntax**

**Definition:**

```
FUNCTION F_VN_GetAt_TcVnVector2_UINT : HRESULT
VAR_INPUT
  ipContainer : ITcVnContainer;
  aElement    : Reference To TcVnVector2_UINT;
  nIndex      : ULINT;
  hrPrev      : HRESULT;
END_VAR
```
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with TcVnVector2_UINT elements</td>
</tr>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector2_UINT [153]</td>
<td>Returns the element at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.4.20  F_VN_GetAt_TcVnVector2_USINT

Gets the TcVnVector2_USINT element at the specified index of the container.

### Syntax

**Definition:**

FUNCTION F_VN_GetAt_TcVnVector2_USINT : HRESULT

VAR_INPUT

- ipContainer : ITcVnContainer;
- aElement : Reference To TcVnVector2_USINT;
- nIndex : ULINT;
- hrPrev : HRESULT;

END_VAR
Return value

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.4.21  F_VN_GetAt_TcVnVector3_INT

F_VN_GetAt_TcVnVector3_INT

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvnContainer</td>
<td>Container with TcVnVector3_INT elements</td>
</tr>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector3_INT</td>
<td>Returns the element at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Gets the TcVnVector3_INT element at the specified index of the container.

Syntax

Definition:

FUNCTION F_VN_GetAt_TcVnVector3_INT : HRESULT
VAR_INPUT
   ipContainer : ITCvnContainer;
   aElement : Reference To TcVnVector3_INT;
   nIndex : ULINT;
   hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvnContainer [345]</td>
<td>Container with TcVnVector3_INT elements</td>
</tr>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector3_INT [153]</td>
<td>Returns the element at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License
TC3 Vision Base
# System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.4.4.22 F_VN_GetAt_TcVnVector3_REAL

![Function Diagram]

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with TcVnVector3_REAL elements</td>
</tr>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector3_REAL</td>
<td>Returns the element at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.4.23 F_VN_GetAt_TcVnVector3_SINT

**F_VN_GetAt_TcVnVector3_SINT**

- ipContainer : ITCvnContainer [345]
- aElement : Reference To TcVnVector3_SINT
- nIndex : ULINT
- hrPrev : HRESULT

Gets the TcVnVector3_SINT element at the specified index of the container.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_GetAt_TcVnVector3_SINT : HRESULT
VAR_INPUT
   ipContainer : ITCvnContainer;
   aElement : Reference To TcVnVector3_SINT;
   nIndex : ULINT;
   hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvnContainer [345]</td>
<td>Container with TcVnVector3_SINT elements</td>
</tr>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector3_SINT [153]</td>
<td>Returns the element at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.24 F_VN_GetAt_TcVnVector3_UINT

**F_VN_GetAt_TcVnVector3_UINT**

- ipContainer : ITCvnContainer [345]
- aElement : Reference To TcVnVector3_UINT
- nIndex : ULINT
- hrPrev : HRESULT

Gets the TcVnVector3_UINT element at the specified index of the container.
**Syntax**

**Definition:**

```c
FUNCTION F_VN_GetAt_TcVnVector3_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aElement    : Reference To TcVnVector3_UINT;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with TcVnVector3_UINT elements</td>
</tr>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector3_UINT</td>
<td>Returns the element at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

**HRESULT [rę 135]**

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.4.4.25 F_VN_GetAt_TcVnVector3_USINT**

```c
F_VN_GetAt_TcVnVector3_USINT
// ipContainer ITcVnContainer
// aElement Reference To TcVnVector3_USINT
// nIndex ULINT
// hrPrev HRESULT
```

Gets the TcVnVector3_USINT element at the specified index of the container.

### Syntax

**Definition:**

```c
FUNCTION F_VN_GetAt_TcVnVector3_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aElement    : Reference To TcVnVector3_USINT;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with TcVnVector3_USINT elements</td>
</tr>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector3_USINT [153]</td>
<td>Returns the element at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.4.4.26 F_VN_GetAt_TcVnVector4_DINT

`F_VN_GetAt_TcVnVector4_DINT` function gets the TcVnVector4_DINT element at the specified index of the container.

### Syntax

**Definition:**

```c
FUNCTION F_VN_GetAt_TcVnVector4_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aElement    : Reference To TcVnVector4_DINT;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

gets the TcVnVector4_DINT element at the specified index of the container.
Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.4.27  F_VN_GetAt_TcVnVector4_INT

F_VN_GetAt_TcVnVector4_INT

<table>
<thead>
<tr>
<th>ipContainer</th>
<th>ITCvnContainer</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector4_INT</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
</tr>
</tbody>
</table>

Gets the TcVnVector4_INT element at the specified index of the container.

Syntax

Definition:

FUNCTION F_VN_GetAt_TcVnVector4_INT : HRESULT
VAR_INPUT
    ipContainer : ITCvnContainer;
    aElement : Reference To TcVnVector4_INT;
    nIndex     : ULINT;
    hrPrev     : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvnContainer [345]</td>
<td>Container with TcVnVector4_INT elements</td>
</tr>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector4_INT [153]</td>
<td>Returns the element at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.4.28 F_VN_GetAt_TcVnVector4_LREAL

F_VN_GetAt_TcVnVector4_LREAL

```
ipContainer : ITcVnContainer;
aElement : Reference To TcVnVector4_LREAL;
nIndex : ULINT;
hrPrev : HRESULT;
```

Gets the TcVnVector4_LREAL element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_TcVnVector4_LREAL : HRESULT
VAR_INPUT
  ipContainer : ITcVnContainer;
  aElement : Reference To TcVnVector4_LREAL;
  nIndex : ULINT;
  hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with TcVnVector4_LREAL elements</td>
</tr>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Returns the element at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.4.29  F_VN_GetAt_TcVnVector4_SINT

gets the TcVnVector4_SINT element at the specified index of the container.

**Syntax**

**Definition:**

FUNCTION F_VN_GetAt_TcVnVector4_SINT : HRESULT
VAR_INPUT
  ipContainer : ITCvntContainer;
  aElement : Reference To TcVnVector4_SINT;
  nIndex : ULINT;
  hrPrev : HRESULT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvntContainer [345]</td>
<td>Container with TcVnVector4_SINT elements</td>
</tr>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector4_SINT</td>
<td>Returns the element at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.30  F_VN_GetAt_TcVnVector4_UINT

gets the TcVnVector4_UINT element at the specified index of the container.
**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_GetAt_TcVnVector4_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aElement    : Reference To TcVnVector4_UINT;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with TcVnVector4_UINT elements</td>
</tr>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector4_UINT</td>
<td>Returns the element at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64)</td>
<td>Tc3_Vision</td>
</tr>
<tr>
<td></td>
<td>with PL50, e.g.</td>
<td>Intel 4-core Atom CPU</td>
</tr>
</tbody>
</table>

6.4.4.4.31 F_VN_GetAt_TcVnVector4_USINT

F_VN_GetAt_TcVnVector4_USINT

```plaintext
F_VN_GetAt_TcVnVector4_USINT
    ipContainer : ITcVnContainer
    aElement    : Reference To TcVnVector4_USINT
    nIndex      : ULINT
    hrPrev      : HRESULT
```

Gets the TcVnVector4_USINT element at the specified index of the container.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_GetAt_TcVnVector4_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aElement    : Reference To TcVnVector4_USINT;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with UDINT elements</td>
</tr>
<tr>
<td>aElement</td>
<td>Reference To UDINT [153]</td>
<td>Returns the element at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.4.32  F_VN_GetAt_UDINT

**F_VN_GetAt_UDINT**

- ipContainer : ITcVnContainer;
- nElement : Reference To UDINT;
- nIndex : ULINT;
- hrPrev : HRESULT;

Gets the UDINT element at the specified index of the container.

### Syntax

**Definition:**

FUNCTION F_VN_GetAt_UDINT : HRESULT

VAR_INPUT

- ipContainer : ITcVnContainer;
- nElement : Reference To UDINT;
- nIndex : ULINT;
- hrPrev : HRESULT;

END_VAR
### Return value

**HRESULT [135]**

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.4.33 F_VN_GetAt_UINT

```vbnet
FUNCTION F_VN_GetAt_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nElement    : Reference To UINT;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Gets the UINT element at the specified index of the container.

### Syntax

**Definition:**

```vbnet
FUNCTION F_VN_GetAt_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nElement    : Reference To UINT;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with UINT elements</td>
</tr>
<tr>
<td>nElement</td>
<td>Reference To UINT</td>
<td>Returns the element at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

**HRESULT [135]**

### Required License

TC3 Vision Base
### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.4.34  F_VN_GetAt_ULINT

#### F_VN_GetAt_ULINT

```plaintext
ipContainer : ITcVnContainer; 
nElement    : Reference To ULINT; 
nIndex       : ULINT; 
hrPrev       : HRESULT;
```

Gets the ULINT element at the specified index of the container.

#### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_GetAt_ULINT : HRESULT 
VAR_INPUT 
   ipContainer : ITcVnContainer;
   nElement    : Reference To ULINT;
   nIndex      : ULINT;
   hrPrev      : HRESULT;
END_VAR 
```

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [_345]</td>
<td>Container with ULINT elements</td>
</tr>
<tr>
<td>nElement</td>
<td>Reference To ULINT</td>
<td>Returns the element at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [_135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

#### Return value

**HRESULT [_135]**

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.4.4.35  F_VN_GetAt_USINT

F_VN_GetAt_USINT

ipContainer  ITCvNContainer  HRESULT  F_VN_GetAt_USINT
nElement    Reference To USINT
nIndex      ULINT
hrPrev      HRESULT

Gets the USINT element at the specified index of the container.

Syntax

Definition:

FUNCTION F_VN_GetAt_USINT : HRESULT
VAR_INPUT
    ipContainer : ITCvNContainer;
    nElement    : Reference To USINT;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer [345]</td>
<td>Container with USINT elements</td>
</tr>
<tr>
<td>nElement</td>
<td>Reference To USINT</td>
<td>Returns the element at the specified index</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.5  F_VN_InsertIntoContainer

This chapter contains functions to insert an element at a specific container position according to the data type.
6.4.4.5.1  **F_VN_InsertIntoContainer_DINT**

Insert an element of type DINT into a container before the specified position.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_InsertIntoContainer_DINT : HRESULT
VAR_INPUT
    nElement : DINT;
    ipContainer : ITCvntContainer;
    nIndex : ULINT;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nElement</td>
<td>DINT</td>
<td>Single element to insert into <code>ipContainer</code></td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvntContainer</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If <code>SUCCEEDED(hrPrev)</code> equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

```
HRESULT [135]
```

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.5.2  **F_VN_InsertIntoContainer_INT**

Insert an element of type INT into a container before the specified position.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_InsertIntoContainer_INT : HRESULT
VAR_INPUT
    nElement : INT;
    ipContainer : ITCvntContainer;
    nIndex : ULINT;
    hrPrev : HRESULT;
END_VAR
```
FUNCTION F_VN_InsertIntoContainer_INT : HRESULT
VAR_INPUT
  nElement    : INT;
  ipContainer : ITcVnContainer;
  nIndex      : ULINT;
  hrPrev      : HRESULT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nElement</td>
<td>INT</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.4.5.3 F_VN_InsertIntoContainer_ITcVnContainer**

```
ipElement  ITcVnContainer
ipContainer ITcVnContainer
nIndex     ULINT
hrPrev     HRESULT
```

Insert an element (or a range of elements) into a container before the specified position.

**Syntax**

**Definition:**

FUNCTION F_VN_InsertIntoContainer_ITcVnContainer : HRESULT
VAR_INPUT
  ipElement    : ITcVnContainer;
  ipContainer  : ITcVnContainer;
  nIndex       : ULINT;
  hrPrev       : HRESULT;
END_VAR
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipElement</td>
<td>ITcVnForwardIterator [345]</td>
<td>Single element to insert into ipContainer or container with several elements</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.5.4 **F_VN_InsertIntoContainer_ITcVnForwardIterator**

Insert an element (represented by an iterator) into a container before the specified position.

**Syntax**

**Definition:**

FUNCTION F_VN_InsertIntoContainer_ITcVnForwardIterator : HRESULT  
VAR_INPUT  
    ipElement : ITcVnForwardIterator;  
    ipContainer : ITcVnContainer;  
    nIndex : ULINT;  
    hrPrev : HRESULT;  
END_VAR
Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.5.5  F_VN_InsertIntoContainer_LREAL

Insert an element of type LREAL into a container before the specified position.

Syntax

Definition:

```c
FUNCTION F_VN_InsertIntoContainer_LREAL : HRESULT
VAR_INPUT
  fElement    : LREAL;
  ipContainer : ITcVnContainer;
  nIndex      : ULINT;
  hrPrev      : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fElement</td>
<td>LREAL</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base
6.4.4.5.6  F_VN_InsertIntoContainer_REAL

Insert an element of type REAL into a container before the specified position.

**Syntax**

**Definition:**

FUNCTION F_VN_InsertIntoContainer_REAL : HRESULT
VAR_INPUT
  fElement : REAL;
  ipContainer : ITCvNContainer;
  nIndex : ULINT;
  hrPrev : HRESULT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fElement</td>
<td>REAL</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

TF7000 - TF7300  Version: 1.3  631
6.4.4.5.7  **F_VN_InsertIntoContainer_SINT**

 Insert an element of type SINT into a container before the specified position.

**Syntax**

**Definition:**

\[
\text{FUNCTION F_VN_InsertIntoContainer_SINT : HRESULT}
\]

\[
\begin{align*}
\text{VAR_INPUT} \quad & \\
\text{nElement} & : \text{SINT}; \\
\text{ipContainer} & : \text{ITcVnContainer}; \\
\text{nIndex} & : \text{ULINT}; \\
\text{hrPrev} & : \text{HRESULT}; \\
\end{align*}
\]

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nElement</td>
<td>SINT</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

**HRESULT [135]**

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.5.8  **F_VN_InsertIntoContainer_TcVnDMatch**

 Insert an element of type TcVnDMatch into a container before the specified position.

**Syntax**

**Definition:**

\[
\text{FUNCTION F_VN_InsertIntoContainer_TcVnDMatch : HRESULT}
\]

\[
\begin{align*}
\text{VAR_INPUT} \quad & \\
\text{stElement} & : \text{Reference To TcVnDMatch}; \\
\text{ipContainer} & : \text{ITcVnContainer}; \\
\text{nIndex} & : \text{ULINT}; \\
\text{hrPrev} & : \text{HRESULT}; \\
\end{align*}
\]
FUNCTION F_VN_InsertIntoContainer_TcVnDMatch : HRESULT
VAR_INPUT
  stElement : Reference To TcVnDMatch;
  ipContainer : ITcVnContainer;
  nIndex : ULINT;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stElement</td>
<td>Reference To TcVnDMatch</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.5.9 F_VN_InsertIntoContainer_TcVnKeyPoint

Insert an element of type TcVnKeyPoint into a container before the specified position.

Syntax

Definition:

FUNCTION F_VN_InsertIntoContainer_TcVnKeyPoint : HRESULT
VAR_INPUT
  stElement : Reference To TcVnKeyPoint;
  ipContainer : ITcVnContainer;
  nIndex : ULINT;
  hrPrev : HRESULT;
END_VAR
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stElement</td>
<td>Reference To TcVnKeyPoint[208]</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer[345]</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT[135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

#### 6.4.4.5.10 F_VN_InsertIntoContainer_TcVnPoint2_DINT

Insert an element of type TcVnPoint2_DINT into a container before the specified position.

### Syntax

**Definition:**

```vbnet
FUNCTION F_VN_InsertIntoContainer_TcVnPoint2_DINT : HRESULT
VAR_INPUT
  aElement : Reference To TcVnPoint2_DINT;
  ipContainer : ITcVnContainer;
  nIndex : ULINT;
  hrPrev : HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnPoint2_DINT[151]</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer[345]</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.5.11 F_VN_InsertIntoContainer_TcVnPoint2_LREAL

Insert an element of type TcVnPoint2_LREAL into a container before the specified position.

Syntax

Definition:

FUNCTION F_VN_InsertIntoContainer_TcVnPoint2_LREAL : HRESULT
VAR_INPUT
  aElement : Reference To TcVnPoint2_LREAL;
  ipContainer : ITcVnContainer;
  nIndex : ULINT;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnPoint2_LREAL [151]</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base
### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.5.12  F_VN_InsertIntoContainer_TcVnPoint2_REAL

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_InsertIntoContainer_TcVnPoint2_REAL : HRESULT
VAR_INPUT
    aElement : Reference To TcVnPoint2_REAL;
    ipContainer : ITcVnContainer;
    nIndex : ULINT;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnPoint2_REAL</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.5.13  F_VN_InsertIntoContainer_TcVnPoint3_LREAL

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_InsertIntoContainer_TcVnPoint3_LREAL : HRESULT
VAR_INPUT
    aElement : Reference To TcVnPoint3_LREAL;
    ipContainer : ITcVnContainer;
    nIndex : ULINT;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnPoint3_LREAL</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]
Insert an element of type TcVnPoint3_LREAL into a container before the specified position.

Syntax

Definition:

```plaintext
FUNCTION F_VN_InsertIntoContainer_TcVnPoint3_LREAL : HRESULT
VAR_INPUT
    aElement    : Reference To TcVnPoint3_LREAL;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnPoint3_LREAL [151]</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.5.14  F_VN_InsertIntoContainer_TcVnPoint3_REAL

Insert an element of type TcVnPoint3_REAL into a container before the specified position.

Syntax

Definition:

```plaintext
FUNCTION F_VN_InsertIntoContainer_TcVnPoint3_REAL : HRESULT
VAR_INPUT
    aElement    : Reference To TcVnPoint3_REAL;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stElement</td>
<td>Reference To TcVnRectangle_DINT</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.5.15 F_VN_InsertIntoContainer_TcVnRectangle_DINT

Insert an element of type TcVnRectangle_DINT into a container before the specified position.

**Syntax**

**Definition:**

FUNCTION F_VN_InsertIntoContainer_TcVnRectangle_DINT : HRESULT
VAR_INPUT
  stElement : Reference To TcVnRectangle_DINT;
  ipContainer : ITcVnContainer;
  nIndex : ULINT;
  hrPrev : HRESULT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stElement</td>
<td>Reference To TcVnRectangle_DINT</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.5.16 F_VN_InsertIntoContainer_TcVnVector2_DINT

Insert an element of type TcVnVector2_DINT into a container before the specified position.

Syntax

Definition:

FUNCTION F_VN_InsertIntoContainer_TcVnVector2_DINT : HRESULT
VAR_INPUT
    aElement : Reference To TcVnVector2_DINT;
    ipContainer : ITCvnContainer;
    nIndex : ULINT;
    hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector2_DINT</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvnContainer</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.5.17 **F_VN_InsertIntoContainer_TcVnVector2_INT**

**Syntax**

**Definition:**

FUNCTION F_VN_InsertIntoContainer_TcVnVector2_INT : HRESULT
VAR_INPUT
   aElement : Reference To TcVnVector2_INT;
   ipContainer : ITcVnContainer;
   nIndex : ULINT;
   hrPrev : HRESULT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector2_INT</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.5.18 **F_VN_InsertIntoContainer_TcVnVector2_REAL**

**Syntax**

**Definition:**

FUNCTION F_VN_InsertIntoContainer_TcVnVector2_REAL : HRESULT
VAR_INPUT
   aElement : Reference To TcVnVector2_REAL;
   ipContainer : ITcVnContainer;
   nIndex : ULINT;
   hrPrev : HRESULT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector2_REAL</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]
Insert an element of type TcVnVector2_REAL into a container before the specified position.

Syntax

Definition:

FUNCTION F_VN_InsertIntoContainer_TcVnVector2_REAL : HRESULT
VAR_INPUT
  aElement    : Reference To TcVnVector2_REAL;
  ipContainer : ITcVnContainer;
  nIndex      : ULINT;
  hrPrev      : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector2_REAL [153]</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.5.19  F_VN_InsertIntoContainer_TcVnVector2_SINT

Insert an element of type TcVnVector2_SINT into a container before the specified position.

Syntax

Definition:

FUNCTION F_VN_InsertIntoContainer_TcVnVector2_SINT : HRESULT
VAR_INPUT
  aElement    : Reference To TcVnVector2_SINT;
  ipContainer : ITcVnContainer;
  nIndex      : ULINT;
  hrPrev      : HRESULT;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector2_SINT [153]</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer [345]</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.4.5.20 F_VN_InsertIntoContainer_TcVnVector2_UINT

Insert an element of type TcVnVector2_UINT into a container before the specified position.

### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_InsertIntoContainer_TcVnVector2_UINT : HRESULT
VAR_INPUT
    aElement     Reference To TcVnVector2_UINT;
    ipContainer  ITCvNContainer;
    nIndex       ULINT;
    hrPrev       HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector2_UINT [153]</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer [345]</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
6.4.4.5.21 F_VN_InsertIntoContainer_TcVnVector2_USINT

Insert an element of type TcVnVector2_USINT into a container before the specified position.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_InsertIntoContainer_TcVnVector2_USINT : HRESULT
VAR_INPUT
    aElement : Reference To TcVnVector2_USINT;
    ipContainer : ITCvContainer;
    nIndex : ULINT;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector2_USINT</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvContainer</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

**HRESULT [135]**

**Required License**

TC3 Vision Base
### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.5.22 F_VN_InsertIntoContainer_TcVnVector3_INT

#### Insert an element of type TcVnVector3_INT into a container before the specified position.

#### Syntax

**Definition:**

```c
FUNCTION F_VN_InsertIntoContainer_TcVnVector3_INT : HRESULT
VAR_INPUT
    aElement : Reference To TcVnVector3_INT;
    ipContainer : ITcVnContainer;
    nIndex : ULINT;
    hrPrev : HRESULT;
END_VAR
```

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector3_INT</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

#### Return value

**HRESULT [135]**

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.5.23 F_VN_InsertIntoContainer_TcVnVector3_REAL

#### Insert an element of type TcVnVector3_REAL into a container before the specified position.

#### Syntax

**Definition:**

```c
FUNCTION F_VN_InsertIntoContainer_TcVnVector3_REAL : HRESULT
VAR_INPUT
    aElement : Reference To TcVnVector3_REAL;
    ipContainer : ITcVnContainer;
    nIndex : ULINT;
    hrPrev : HRESULT;
END_VAR
```
Insert an element of type TcVnVector3_REAL into a container before the specified position.

Syntax

Definition:

FUNCTION F_VN_InsertIntoContainer_TcVnVector3_REAL : HRESULT
VAR_INPUT
  aElement : Reference To TcVnVector3_REAL;
  ipContainer : ITcVnContainer;
  nIndex : ULINT;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector3_REAL</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.5.24  F_VN_InsertIntoContainer_TcVnVector3_SINT

Insert an element of type TcVnVector3_SINT into a container before the specified position.

Syntax

Definition:

FUNCTION F_VN_InsertIntoContainer_TcVnVector3_SINT : HRESULT
VAR_INPUT
  aElement : Reference To TcVnVector3_SINT;
  ipContainer : ITcVnContainer;
  nIndex : ULINT;
  hrPrev : HRESULT;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector3_UINT</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.4.5.25 F_VN_InsertIntoContainer_TcVnVector3_UINT

Insert an element of type TcVnVector3_UINT into a container before the specified position.

### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_InsertIntoContainer_TcVnVector3_UINT : HRESULT
VAR_INPUT
    aElement : Reference To TcVnVector3_UINT;
    ipContainer : ITcVnContainer;
    nIndex : ULINT;
    hrPrev : HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector3_UINT</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.5.26 F_VN_InsertIntoContainer_TcVnVector3_USINT

Insert an element of type TcVnVector3_USINT into a container before the specified position.

**Syntax**

**Definition:**

FUNCTION F_VN_InsertIntoContainer_TcVnVector3_USINT : HRESULT
VAR_INPUT
  aElement : Reference To TcVnVector3_USINT;
  ipContainer : ITCvContainer;
  nIndex : ULINT;
  hrPrev : HRESULT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector3_USINT [153]</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvContainer [345]</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base
6.4.4.5.27  F_VN_InsertIntoContainer_TcVnVector4_DINT

**Insert an element of type TcVnVector4_DINT into a container before the specified position.**

**Syntax**

**Definition:**

FUNCTION `F_VN_InsertIntoContainer_TcVnVector4_DINT` : HRESULT

VAR_INPUT

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector4_DINT</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvnContainer</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Inputs**

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
Insert an element of type TcVnVector4_INT into a container before the specified position.

**Syntax**

**Definition:**

```
FUNCTION F_VN_InsertIntoContainer_TcVnVector4_INT : HRESULT
VAR_INPUT
  aElement : Reference To TcVnVector4_INT;
  ipContainer : ITCvNContainer;
  nIndex : ULINT;
  hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector4_INT</td>
<td>Single element to insert into ipContainer [153]</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer</td>
<td>Container in which to insert the element [345]</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.5.29 **F_VN_InsertIntoContainer_TcVnVector4_LREAL**

Insert an element of type TcVnVector4_LREAL into a container before the specified position.

**Syntax**

**Definition:**

```
FUNCTION F_VN_InsertIntoContainer_TcVnVector4_LREAL : HRESULT
VAR_INPUT
  aElement : Reference To TcVnVector4_LREAL;
  ipContainer : ITCvNContainer;
  nIndex : ULINT;
  hrPrev : HRESULT;
END_VAR
```
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.5.30  F_VN_InsertIntoContainer_TcVnVector4_SINT

Insert an element of type TcVnVector4_SINT into a container before the specified position.

Syntax

Definition:

FUNCTION F_VN_InsertIntoContainer_TcVnVector4_SINT : HRESULT
VAR_INPUT
  aElement : Reference To TcVnVector4_SINT;
  ipContainer : ITcVnContainer;
  nIndex : ULINT;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector4_SINT [153]</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.5.31 F_VN_InsertIntoContainer_TcVnVector4_UINT

**Syntax**

Definition:

FUNCTION F_VN_InsertIntoContainer_TcVnVector4_UINT : HRESULT

VAR_INPUT

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector4_UINT</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Inputs**

**Return value**

HRESULT [135]

Required License
TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.5.32 F_VN_InsertIntoContainer_TcVnVector4_USINT

**F_VN_InsertIntoContainer_TcVnVector4_USINT**

Insert an element of type TcVnVector4_USINT into a container before the specified position.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_InsertIntoContainer_TcVnVector4_USINT : HRESULT
VAR_INPUT
    aElement : Reference To TcVnVector4_USINT;
    ipContainer : ITcVnContainer;
    nIndex : ULINT;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector4_USINT [153]</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

**HRESULT [135]**

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.4.5.33 F_VN_InsertIntoContainer_UDINT

Insert an element of type UDINT into a container before the specified position.

Syntax

Definition:

FUNCTION F_VN_InsertIntoContainer_UDINT : HRESULT
VAR_INPUT
  nElement : UDINT;
  ipContainer : ITcVnContainer;
  nIndex : ULINT;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nElement</td>
<td>UDINT</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
FUNCTION F_VN_InsertIntoContainer_UINT : HRESULT
VAR_INPUT
    nElement : UINT;
    ipContainer : ITcVnContainer;
    nIndex : ULINT;
    hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nElement</td>
<td>UINT</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.5.35  F_VN_InsertIntoContainer_ULINT

Insert an element of type ULINT into a container before the specified position.

Syntax

Definition:
FUNCTION F_VN_InsertIntoContainer_ULINT : HRESULT
VAR_INPUT
    nElement : ULINT;
    ipContainer : ITcVnContainer;
    nIndex : ULINT;
    hrPrev : HRESULT;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nElement</td>
<td>ULINT</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer [345]</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.5.36  **F_VN_InsertIntoContainer_USINT**

Insert an element of type USINT into a container before the specified position.

### Syntax

**Definition:**

```vb
FUNCTION F_VN_InsertIntoContainer_USINT : HRESULT
VAR_INPUT
    nElement    : USINT;
    ipContainer : ITCvNContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nElement</td>
<td>USINT</td>
<td>Single element to insert into ipContainer</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer [345]</td>
<td>Container in which to insert the element</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Position, before which the element will be inserted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.6 F_VN_SetAt

The F_VN_SetAt functions should be used if individual elements in a container have to be accessed. If all elements in a container are to be processed in succession, it is much better to use an iterator [706].

6.4.4.6.1 F_VN_SetAt_DINT

Sets the element at the specified index of the container.

Syntax

Definition:

FUNCTION F_VN_SetAt_DINT : HRESULT
VAR_INPUT
  nElement : DINT;
  ipContainer : ITcVnContainer;
  nIndex : ULINT;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nElement</td>
<td>DINT</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with DINT elements, in which the element at position nIndex is replaced by nElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]
6.4.4.6.2  F_VN_SetAt_INT

Sets the element at the specified index of the container.

Syntax

Definition:

```plaintext
FUNCTION F_VN_SetAt_INT : HRESULT
VAR_INPUT
  nElement : INT;
  ipContainer : ITcVnContainer;
  nIndex : ULINT;
  hrPrev : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nElement</td>
<td>INT</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with INT elements, in which the element at position nIndex is replaced by nElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.4.6.3  F_VN_SetAt_ITcVnContainer

Sets the element at the specified index of the container. (Alternatively use interface method .SetAt.)

Syntax

Definition:
FUNCTION F_VN_SetAt_ITcVnContainer : HRESULT
VAR_INPUT
  ipElement   : ITcVnContainer;
  ipContainer : ITcVnContainer;
  nIndex      : ULINT;
  hrPrev      : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipElement</td>
<td>ITcVnContainer</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with ITcVnContainer elements, in which the element at position nIndex is replaced by ipElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.6.4  F_VN_SetAt_LREAL

Sets the element at the specified index of the container.
Syntax

Definition:

```plaintext
FUNCTION F_VN_SetAt_REAL : HRESULT
VAR_INPUT
    fElement    : REAL;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fElement</td>
<td>LREAL</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with REAL elements, in which the element at position nIndex is replaced by fElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-Core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.6.5 F_VN_SetAt_REAL

```plaintext
F_VN_SetAt_REAL
  fElement REAL
  ipContainer ITcVnContainer
  nIndex ULINT
  hrPrev HRESULT

Sets the element at the specified index of the container.
```

## Syntax

Definition:

```plaintext
FUNCTION F_VN_SetAt_REAL : HRESULT
VAR_INPUT
    fElement    : REAL;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nElement</td>
<td>SINT</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with SINT elements, in which the element at position nIndex is replaced by nElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.6.6 F_VN_SetAt_SINT

Sets the element at the specified index of the container.

#### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_SetAt_SINT : HRESULT
VAR_INPUT
  nElement : SINT;
  ipContainer : ITcVnContainer;
  nIndex : ULINT;
  hrPrev : HRESULT;
END_VAR
```

## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nElement</td>
<td>SINT</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with SINT elements, in which the element at position nIndex is replaced by nElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.6.7  F_VN_SetAt_TcVnDMatch

Sets the element at the specified index of the container.

#### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_SetAt_TcVnDMatch : HRESULT
VAR_INPUT
  stElement : Reference To TcVnDMatch;
  ipContainer : ITCvNContainer;
  nIndex : ULINT;
  hrPrev : HRESULT;
END_VAR
```

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stElement</td>
<td>Reference To TcVnDMatch</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer</td>
<td>Container with TcVnDMatch elements, in which the element at position nIndex is replaced by stElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base
6.4.4.6.8  F_VN_SetAt_TcVnKeyPoint

Sets the element at the specified index of the container.

Syntax

Definition:

FUNCTION F_VN_SetAt_TcVnKeyPoint : HRESULT
VAR_INPUT
    stElement  : Reference To TcVnKeyPoint;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stElement</td>
<td>Reference To TcVnKeyPoint</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with TcVnKeyPoint elements, in which the element at position nIndex is replaced by stElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.4.6.9 F_VN_SetAt_TcVnPoint2_DINT

Sets the element at the specified index of the container.

Syntax

Definition:

FUNCTION F_VN_SetAt_TcVnPoint2_DINT : HRESULT
VAR_INPUT
  aElement : Reference To TcVnPoint2_DINT;
  ipContainer : ITCvNContainer;
  nIndex : ULINT;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnPoint2_DINT</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer</td>
<td>Container with TcVnPoint2_DINT elements, in which the element at position nIndex is replaced by aElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.6.10 F_VN_SetAt_TcVnPoint2_LREAL

Sets the element at the specified index of the container.

FUNCTION F_VN_SetAt_TcVnPoint2_LREAL : HRESULT
VAR_INPUT
  aElement : Reference To TcVnPoint2_LREAL;
  ipContainer : ITCvNContainer;
  nIndex : ULINT;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnPoint2_LREAL</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer</td>
<td>Container with TcVnPoint2_LREAL elements, in which the element at position nIndex is replaced by aElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_SetAt_TcVnPoint2_LREAL : HRESULT
VAR_INPUT
    aElement    : Reference To TcVnPoint2_LREAL;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnPoint2_LREAL [151]</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with TcVnPoint2_LREAL elements, in which the element at position nIndex is replaced by aElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.4.6.11 F_VN_SetAt_TcVnPoint2_REAL**

Sets the element at the specified index of the container.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_SetAt_TcVnPoint2_REAL : HRESULT
VAR_INPUT
    aElement    : Reference To TcVnPoint2_REAL;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnPoint2_REAL</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with TcVnPoint2_REAL elements, in which the element at position nIndex is replaced by aElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.6.12 F_VN_SetAt_TcVnPoint3_LREAL

Sets the element at the specified index of the container.

#### Syntax

**Definition:**

```c
FUNCTION F_VN_SetAt_TcVnPoint3_LREAL : HRESULT
VAR_INPUT
  aElement : Reference To TcVnPoint3_LREAL;
  ipContainer : ITcVnContainer;
  nIndex : ULINT;
  hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnPoint3_LREAL</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with TcVnPoint3_LREAL elements, in which the element at position nIndex is replaced by aElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.6.13  **F_VN_SetAt_TcVnPoint3_REAL**

<table>
<thead>
<tr>
<th>aElement</th>
<th>Reference To TcVnPoint3_REAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
</tr>
</tbody>
</table>

Sets the element at the specified index of the container.

Syntax

**Definition:**

```
FUNCTION F_VN_SetAt_TcVnPoint3_REAL : HRESULT
VAR_INPUT
    aElement : Reference To TcVnPoint3_REAL;
    ipContainer : ITcVnContainer;
    nIndex : ULINT;
    hrPrev : HRESULT;
END_VAR
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnPoint3_REAL [151]</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with TcVnPoint3_REAL elements, in which the element at position nIndex is replaced by aElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.6.14  F_VN_SetAt_TcVnRectangle_DINT

```
F_VN_SetAt_TcVnRectangle_DINT

stElement : Reference To TcVnRectangle_DINT;
ipContainer : ITcVnContainer;
nIndex : ULINT;
hrPrev : HRESULT;

RESULT : F_VN_SetAt_TcVnRectangle_DINT
```

Sets the element at the specified index of the container.

Syntax

Definition:

```c
FUNCTION F_VN_SetAt_TcVnRectangle_DINT : HRESULT
VAR_INPUT
    stElement : Reference To TcVnRectangle_DINT;
    ipContainer : ITcVnContainer;
    nIndex : ULINT;
    hrPrev : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stElement</td>
<td>Reference To TcVnRectangle_DINT</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with TcVnRectangle_DINT elements, in which the element at position nIndex is replaced by stElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.4.6.15  F_VN_SetAt_TcVnVector2_DINT

Sets the element at the specified index of the container.

Syntax

Definition:
FUNCTION F_VN_SetAt_TcVnVector2_DINT : HRESULT
VAR_INPUT
  aElement  : Reference To TcVnVector2_DINT;
  ipContainer : ITcVnContainer;
  nIndex     : ULINT;
  hrPrev     : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector2_DINT</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with TcVnVector2_DINT elements, in which the element at position nIndex is replaced by aElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.6.16  F_VN_SetAt_TcVnVector2_INT

Sets the element at the specified index of the container.
Syntax

Definition:

FUNCTION F_VN_SetAt_TcVnVector2_INT : HRESULT
VAR_INPUT
    aElement : Reference To TcVnVector2_INT;
    ipContainer : ITcVnContainer;
    nIndex : ULINT;
    hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector2_INT [153]</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with TcVnVector2_INT elements, in which the element at position nIndex is replaced by aElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.6.4.17  F_VN_SetAt_TcVnVector2_REAL

F_VN_SetAt_TcVnVector2_REAL

aElement  Reference To TcVnVector2_REAL
ipContainer ITcVnContainer
nIndex    ULINT
hrPrev    HRESULT

Sets the element at the specified index of the container.

Syntax

Definition:

FUNCTION F_VN_SetAt_TcVnVector2_REAL : HRESULT
VAR_INPUT
    aElement : Reference To TcVnVector2_REAL;
    ipContainer : ITcVnContainer;
    nIndex : ULINT;
    hrPrev : HRESULT;
END_VAR
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector2_REAL</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer</td>
<td>Container with TcVnVector2_REAL elements, in which the element at position nIndex is replaced by aElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.6.18  F_VN_SetAt_TcVnVector2_SINT

Sets the element at the specified index of the container.

Syntax

Definition:

```c
FUNCTION F_VN_SetAt_TcVnVector2_SINT : HRESULT
VAR_INPUT
  aElement : Reference To TcVnVector2_SINT;
  ipContainer : ITCvNContainer;
  nIndex : ULINT;
  hrPrev : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector2_SINT</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer</td>
<td>Container with TcVnVector2_SINT elements, in which the element at position nIndex is replaced by aElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.6.19  F_VN_SetAt_TcVnVector2_UINT

Sets the element at the specified index of the container.

Syntax

Definition:

FUNCTION F_VN_SetAt_TcVnVector2_UINT : HRESULT
VAR_INPUT
  aElement : Reference To TcVnVector2_UINT;
  ipContainer : ITCvnnContainer;
  nIndex : ULINT;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector2_UINT [153]</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvnnContainer [345]</td>
<td>Container with TcVnVector2_UINT elements, in which the element at position nIndex is replaced by aElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base
6.4.4.6.20  **F_VN_SetAt_TcVnVector2_USINT**

Sets the element at the specified index of the container.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_SetAt_TcVnVector2_USINT : HRESULT
VAR_INPUT
    aElement : Reference To TcVnVector2_USINT;
    ipContainer : ITcVnContainer;
    nIndex : ULINT;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector2_USINT</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with TcVnVector2_USINT elements, in which the element at position nIndex is replaced by aElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

**HRESULT [135]**

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.4.6.21   F_VN_SetAt_TcVnVector3_INT

Sets the element at the specified index of the container.

Syntax

Definition:

FUNCTION F_VN_SetAt_TcVnVector3_INT : HRESULT
VAR_INPUT
  aElement : Reference To TcVnVector3_INT;
  ipContainer : ITcVnContainer;
  nIndex : ULINT;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector3_INT</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with TcVnVector3_INT elements, in which the element at position nIndex is replaced by aElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.6.22   F_VN_SetAt_TcVnVector3_REAL

Sets the element at the specified index of the container.
Syntax

Definition:

FUNCTION F_VN_SetAt_TcVnVector3_REAL : HRESULT
VAR_INPUT
    aElement : Reference To TcVnVector3_REAL;
    ipContainer : ITcVnContainer;
    nIndex : ULINT;
    hrPrev : HRESULT;
END_VAR

## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector3_REAL</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with TcVnVector3_REAL elements, in which the element at position nIndex is replaced by aElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.6.23 F_VN_SetAt_TcVnVector3_SINT

Sets the element at the specified index of the container.

**Syntax**

**Definition:**

FUNCTION F_VN_SetAt_TcVnVector3_SINT : HRESULT
VAR_INPUT
    aElement : Reference To TcVnVector3_SINT;
    ipContainer : ITcVnContainer;
    nIndex : ULINT;
    hrPrev : HRESULT;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector3_UINT</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with TcVnVector3_UINT elements, in which the element at position nIndex is replaced by aElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## Syntax

**Definition:**

FUNCTION F_VN_SetAt_TcVnVector3_UINT : HRESULT  
VAR_INPUT
  aElement : Reference To TcVnVector3_UINT;  
  ipContainer : ITcVnContainer;  
  nIndex : ULINT;  
  hrPrev : HRESULT;
END_VAR

Sets the element at the specified index of the container.

**Syntax**

FUNCTION F_VN_SetAt_TcVnVector3_UINT : HRESULT  
VAR_INPUT
  aElement : Reference To TcVnVector3_UINT;  
  ipContainer : ITcVnContainer;  
  nIndex : ULINT;  
  hrPrev : HRESULT;
END_VAR

Sets the element at the specified index of the container.

## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector3_UINT</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with TcVnVector3_UINT elements, in which the element at position nIndex is replaced by aElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

6.4.4.6.24 **F_VN_SetAt_TcVnVector3_UINT**

Sets the element at the specified index of the container.
Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.6.25 F_VN_SetAt_TcVnVector3_USINT

F_VN_SetAt_TcVnVector3_USINT

- aElement: Reference To TcVnVector3_USINT
- ipContainer: ITcVnContainer
- nIndex: ULINT
- hrPrev: HRESULT

Sets the element at the specified index of the container.

Syntax

Definition:

FUNCTION F_VN_SetAt_TcVnVector3_USINT : HRESULT

VAR_INPUT

  aElement : Reference To TcVnVector3_USINT;
  ipContainer : ITcVnContainer;
  nIndex : ULINT;
  hrPrev : HRESULT;

END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector3_USINT [153]</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with TcVnVector3_USINT elements, in which the element at position nIndex is replaced by aElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.6.26  **F_VN_SetAt_TcVnVector4_DINT**

Sets the element at the specified index of the container.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_SetAt_TcVnVector4_DINT : HRESULT
VAR_INPUT
   aElement : Reference To TcVnVector4_DINT;
   ipContainer : ITcVnContainer;
   nIndex : ULINT;
   hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector4_DINT</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with TcVnVector4_DINT elements, in which the element at position nIndex is replaced by aElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

**HRESULT**

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.4.6.27  F_VN_SetAt_TcVnVector4_INT

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F_VN_SetAt_TcVnVector4_INT</td>
<td>Sets the element at the specified index of the container.</td>
</tr>
</tbody>
</table>

### Syntax

**Definition:**

```c
FUNCTION F_VN_SetAt_TcVnVector4_INT : HRESULT
VAR_INPUT
    aElement : Reference To TcVnVector4_INT;
    ipContainer : ITcVnContainer;
    nIndex : ULINT;
    hrPrev : HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector4_INT</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with TcVnVector4_INT elements, in which the element at position nIndex is replaced by aElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

**HRESULT [135]**

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.6.28  F_VN_SetAt_TcVnVector4_LREAL

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F_VN_SetAt_TcVnVector4_LREAL</td>
<td>Sets the element at the specified index of the container.</td>
</tr>
</tbody>
</table>
Syntax

Definition:

```c
FUNCTION F_VN_SetAt_TcVnVector4_LREAL : HRESULT
VAR_INPUT
    aElement    : Reference To TcVnVector4_LREAL;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with TcVnVector4_LREAL elements, in which the element at position nIndex is replaced by aElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.6.29 F_VN_SetAt_TcVnVector4_SINT

Sets the element at the specified index of the container.

Syntax

Definition:

```c
FUNCTION F_VN_SetAt_TcVnVector4_SINT : HRESULT
VAR_INPUT
    aElement    : Reference To TcVnVector4_SINT;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector4_UINT [153]</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with TcVnVector4_UINT elements, in which the element at position nIndex is replaced by aElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.6.30  F_VN_SetAt_TcVnVector4_UINT

Sets the element at the specified index of the container.

Syntax

Definition:

FUNCTION F_VN_SetAt_TcVnVector4_UINT : HRESULT
VAR_INPUT
  aElement : Reference To TcVnVector4_UINT;
  ipContainer : ITcVnContainer;
  nIndex : ULINT;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector4_UINT [153]</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with TcVnVector4_UINT elements, in which the element at position nIndex is replaced by aElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
### 6.4.4.6.31 F_VN_SetAt_TcVnVector4_USINT

**Sets the element at the specified index of the container.**

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_SetAt_TcVnVector4_USINT : HRESULT
VAR_INPUT
    aElement : Reference To TcVnVector4_USINT;
    ipContainer : ITCvnContainer;
    nIndex : ULINT;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aElement</td>
<td>Reference To TcVnVector4_USINT</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvnContainer</td>
<td>Container with TcVnVector4_USINT elements, in which the element at position nIndex is replaced by aElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

**HRESULT [135]**

**Required License**

TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.6.32 **F_VN_SetAt_UDINT**

Sets the element at the specified index of the container.

Syntax

**FUNCTION F_VN_SetAt_UDINT : HRESULT**

VAR_INPUT

- **nElement : UDINT**
- **ipContainer : ITCvncContainer**
- **nIndex : ULINT**
- **hrPrev : HRESULT**

END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nElement</td>
<td>UDINT</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITCvncContainer</td>
<td>Container with UDINT elements, in which the element at position nIndex is replaced by nElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.4.6.33  F_VN_SetAt_UINT

Sets the element at the specified index of the container.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_SetAt_UINT : HRESULT
VAR_INPUT
    nElement : UINT;
    ipContainer : ITcVnContainer;
    nIndex : ULINT;
    hrPrev : HRESULT;
END_VAR

RETURN
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nElement</td>
<td>UINT</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with UINT elements, in which the element at position nIndex is replaced by nElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

`HRESULT` [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.6.34  F_VN_SetAt_ULINT

Sets the element at the specified index of the container.
# Syntax

**Definition:**

FUNCTION F_VN_SetAt_ULINT : HRESULT
VAR_INPUT
  nElement : ULINT;
  ipContainer : ITcVnContainer;
  nIndex : ULINT;
  hrPrev : HRESULT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nElement</td>
<td>ULINT</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with ULINT elements, in which the element at position njIndex is replaced by nElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.4.6.35 F_VN_SetAt_USINT

Sets the element at the specified index of the container.

**Syntax**

**Definition:**

FUNCTION F_VN_SetAt_USINT : HRESULT
VAR_INPUT
  nElement : USINT;
  ipContainer : ITcVnContainer;
  nIndex : ULINT;
  hrPrev : HRESULT;
END_VAR
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nElement</td>
<td>USINT</td>
<td>Element to set at the specified container position</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container with USINT elements, in which the element at position nIndex is replaced by nElement</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.7 F_VN_AdvanceIterator

Advance an iterator by the specified offset.

Syntax

Definition:

```
FUNCTION F_VN_AdvanceIterator : HRESULT
VAR_INPUT
  ipIterator : ITcVnForwardIterator;
  nOffset : LINT;
  hrPrev : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipIterator</td>
<td>ITcVnForwardIterator [336]</td>
<td>Iterator to be advanced</td>
</tr>
<tr>
<td>nOffset</td>
<td>LINT</td>
<td>Offset, by which ipIterator is advanced</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
**Return value**

HRESULT [135]

**Further information**

The function F_VN_AdvanceIterator increments or decrements an iterator [334] by a certain offset nOffset.

If the offset nOffset is always to be 1, the simpler function F_VN_IncrementIterator [708] can also be used.

**Parameter**

**Iterator**

The parameter ipIterator is of the type ITCvNForwardIterator. Since ITCvNBidirectionalIterator and ITCvNRandomAccessIterator inherit from it, these can also be transferred to the function. If a forward iterator is to be decremented, an internal attempt is made to transform it into a bidirectional iterator and to decrement it.

**Offset**

The offset nOffset defines by how much the iterator ipIterator is to be incremented or decremented. If nOffset > 0 it is incremented; if nOffset < 0 it is decremented.

**Related functions**

- F_VN_IncrementIterator [708]
- F_VN_AdvanceIterator [685]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.4.8 F_VN_CheckIfEmpty**

```
FUNCTION F_VN_CheckIfEmpty : HRESULT
VAR_INPUT
    ipContainer : ITCvNContainer;
    hrPrev      : HRESULT;
END_VAR
```

Checks if the container is empty. (Alternatively use interface method .CheckIfEmpty.)

**Syntax**

**Definition:**

FUNCTION F_VN_CheckIfEmpty : HRESULT
VAR_INPUT
    ipContainer : ITCvNContainer;
    hrPrev      : HRESULT;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.4.9 F_VN_CheckIfIteratorIsAtEnd

Checks if the iterator points to the past-the-end element. (Alternatively use interface method .CheckIfEnd.)

### Syntax

**Definition:**

```c
definition:            
FUNCTION F_VN_CheckIfIteratorIsAtEnd : HRESULT  
VAR_INPUT            
   ipIterator : ITcVnForwardIterator;  
   hrPrev : HRESULT;  
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipIterator</td>
<td>ITcVnForwardIterator [336]</td>
<td>Iterator</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base
6.4.4.10  

**F_VN_ConvertContainerType**

Converts a container to another type (Struct element types are not supported).

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_ConvertContainerType : HRESULT
VAR_INPUT
    ipSrcContainer : ITCvNContainer;
    ipDestContainer : Reference To ITCvNContainer;
    nDestTypeGuid : GUID;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcContainer</td>
<td>ITCvNContainer ![345]</td>
<td>Source container</td>
</tr>
<tr>
<td>ipDestContainer</td>
<td>Reference To ITCvNContainer ![345]</td>
<td>Returns the converted container</td>
</tr>
<tr>
<td>nDestTypeGuid</td>
<td>GUID</td>
<td>Specifies the destination container type for the conversion</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT ![135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

**HRESULT ![135]**

**Further information**

The function `F_VN_ConvertContainerType` converts a container into a container type ![151] that is to be specified.

The global `ContainerType ![151]` constants can be used to determine the container type via the parameter `nDestTypeGuid`.

**HRESULT**

| 70E   | INCOMPATIBLE                      | The same interface pointer is used as the input container and result container, or a container that has specified a different type to that in `nDestTypeGuid` is used as an interface pointer. |
Application

A typical use case, for example, is the conversion of a container so that it can be drawn in an image. A contour of the type `ContainerType_Vector_TcVnPoint2_REAL`, as is returned among others by the function `F_VN_LocateEllipseExp[{1244}`], can be converted into the type `ContainerType_Vector_TcVnPoint2_DINT` in order to be able to draw it in an image using the function `F_VN_DrawContours[{890}`]:

```c
hr := F_VN_LocateEllipseExp(ipImage, stEllipse, [...], ipContourPoints, hr);
hr := F_VN_ConvertContainerType(
    ipSrcContainer := ipContourPoints,
    ipDestContainer := ipContourToDraw,
    nDestTypeGuid := ContainerType_Vector_TcVnPoint2_DINT,
    hr
);
hr := F_VN_DrawContours(ipContourToDraw, -1, ipImageDisp, aBlue, 5, hr);
```

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.11 F_VN_CopyContainer

Copys a container.

Syntax

Definition:

```c
FUNCTION F_VN_CopyContainer : HRESULT
VAR_INPUT
    ipSrcContainer : ITCvNContainer;
    ipDestContainer : Reference To ITCvNContainer;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcContainer</td>
<td>ITCvNContainer[{345}]</td>
<td>Source container</td>
</tr>
<tr>
<td>ipDestContainer</td>
<td>Reference To ITCvNContainer[{345}]</td>
<td>Destination container (same type ID as ipSrcContainer; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[{135}]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

```c
HRESULT[{135}]
```
Further information

The function \texttt{F\_VN\_CopyContainer} not only copies the \texttt{interface pointer} of the container, it also makes a deep copy of the container.

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

64.4.12 \texttt{F\_VN\_CopyContainerElementsConditional\_ITcVnContainer}

Copys container elements to a new container, depending on a custom condition.

Syntax

Definition:

\begin{verbatim}
FUNCTION F_VN_CopyContainerElementsConditional_ITcVnContainer : HRESULT
VAR_INPUT
  ipSrcContainer  : ITCvNContainer;
  ipDestContainer : Reference To ITCvNContainer;
  ipConditionFB   : ITCvNCustomElementCondition_ITcVnContainer;
  hrPrev           : HRESULT;
END_VAR
\end{verbatim}

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcContainer</td>
<td>ITCvNContainer [1345]</td>
<td>Source container</td>
</tr>
<tr>
<td>ipDestContainer</td>
<td>Reference To ITCvNContainer [1345]</td>
<td>Returns a container with elements, that match the condition</td>
</tr>
<tr>
<td>ipConditionFB</td>
<td>ITCvNCustomElementCondition_ITcVnContainer [226]</td>
<td>Custom condition</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

\begin{verbatim}
HRESULT [135]
\end{verbatim}

Further information

The function \texttt{F\_VN\_CopyContainerElementsConditional} filters elements of multi-dimensional containers [143]. The filter condition is to be specified by a function block with the interface \texttt{ITCvNCustomElementCondition_ITcVnContainer [226]}. 
Parameter

Input container

The input container `ipSrcContainer` must be a multi-dimensional container; i.e. the container must contain sub-containers as elements. Thus all containers of the type `ContainerType_Vector_Vector_...` can be used.

Result container

The result container `ipDestContainer` contains all sub-containers that satisfy the condition in `ipConditionFB`.

Filter condition

The filter condition must be defined as the method `Condition` of a function block. So that this function block can be transferred to the function as the parameter `ipConditionFB`, it must implement the interface `ITcVnCustomElementCondition_ITcVnContainer` [226].

Samples

- [Selections of container elements](#)

Related functions

- F_VN_CopyContainerElementsConditional_ITcVnContainer [690]
- F_VN_CopyContainerElementsConditional_ITcVnForwardIterator [691]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.13 F_VN_CopyContainerElementsConditional_ITcVnForwardIterator

Copys container elements to a new container, depending on a custom condition.

Syntax

Definition:

```c
FUNCTION F_VN_CopyContainerElementsConditional_ITcVnForwardIterator : HRESULT
VAR_INPUT
    ipSrcContainer : ITcVnContainer;
    ipDestContainer : Reference To ITcVnContainer;
    ipConditionFB : ITcVnCustomElementCondition_ITcVnForwardIterator;
    hrPrev : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcContainer</td>
<td>ITCvNContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>ipDestContainer</td>
<td>Reference To ITCvNContainer [345]</td>
<td>Returns a container with elements, that match the condition</td>
</tr>
<tr>
<td>ipConditionFB</td>
<td>ITCvNCustomElementCondition ITCvNForwardIterator [227]</td>
<td>Custom condition</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Further information

The function F_VN_CopyContainerElementsConditional ITCvNForwardIterator filters elements of multi-dimensional containers [143]. The filter condition is to be specified by a function block with the interface ITCvNCustomElementCondition ITCvNForwardIterator [227].

Unlike F_VN_CopyContainerElementsConditional ITCvNContainer [690], sub-containers are not transferred to the conditional method Condition, but iterators, which each point to the first element of the sub-container.

### Parameter

#### Input container

The input container ipSrcContainer must be a multi-dimensional container; i.e. the container must contain sub-containers as elements. Thus all containers of the type ContainerType Vector Vector<...> can be used.

#### Result container

The result container ipDestContainer contains all sub-containers that satisfy the condition in ipConditionFB.

#### Filter condition

The filter condition must be defined as the method Condition of a function block. So that this function block can be transferred to the function as the parameter ipConditionFB, it must implement the interface ITCvNCustomElementCondition ITCvNForwardIterator [227].

### Samples

- Selections of container elements [1364]

### Related functions

- F_VN_CopyContainerElementsConditional ITCvNContainer [690]
- F_VN_CopyContainerElementsConditional ITCvNForwardIterator [691]

### Required License

TC3 Vision Base
### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.14 F_VN_CreateContainer

**Syntax**

**Definition:**

FUNCTION F_VN_CreateContainer : HRESULT

VAR_INPUT

- ipContainer : Reference To ITcVnContainer
- nTypeGuid : GUID
- nElementNum : ULINT
- hrPrev : HRESULT

END_VAR

Creates a container with type GUID nTypeGuid and return its container interface.

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>Reference To ITcVnContainer</td>
<td>Returns the created container</td>
</tr>
<tr>
<td>nTypeGuid</td>
<td>GUID</td>
<td>Type GUID of the container to be created</td>
</tr>
<tr>
<td>nElementNum</td>
<td>ULINT</td>
<td>Number of elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Further information**

The global ContainerType [151] constants can be used to determine the container type via the parameter nTypeGuid.

**Required License**

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.4.15  F_VN_CreateContainerFromArray

**F_VN_CreateContainerFromArray**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pData</td>
<td>PVOID</td>
<td>Pointer to the data array. Make sure that the array contains at least nElementNum elements and that the array element type matches the container element type.</td>
</tr>
<tr>
<td>ipContainer</td>
<td>Reference To ITCvNContainer</td>
<td>Returns the created container</td>
</tr>
<tr>
<td>nTypeGuid</td>
<td>GUID</td>
<td>Type GUID of the container to be created</td>
</tr>
<tr>
<td>nElementNum</td>
<td>ULINT</td>
<td>Number of elements to copy from pData</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Creates a container with type GUID nTypeGuid, initialize it with provided data and return its container interface. Only supported for basic container types, i.e. not for containers of containers.

**Syntax**

**Definition:**

FUNCTION F_VN_CreateContainerFromArray : HRESULT

VAR_INPUT
- pData : PVOID;
- ipContainer : Reference To ITCvNContainer;
- nTypeGuid : GUID;
- nElementNum : ULINT;
- hrPrev : HRESULT;

END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pData</td>
<td>PVOID</td>
<td>Pointer to the data array. Make sure that the array contains at least nElementNum elements and that the array element type matches the container element type.</td>
</tr>
<tr>
<td>ipContainer</td>
<td>Reference To ITCvNContainer</td>
<td>Returns the created container</td>
</tr>
<tr>
<td>nTypeGuid</td>
<td>GUID</td>
<td>Type GUID of the container to be created</td>
</tr>
<tr>
<td>nElementNum</td>
<td>ULINT</td>
<td>Number of elements to copy from pData</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Further information**

The global ContainerType [151] constants can be used to determine the container type via the parameter nTypeGuid.

Since the array data is only specified via a pointer, a type check within the function is not possible. Therefore, please note the following warning:

**WARNING**

**Memory access**

It must be ensured that the type and length of the container correspond to the size of the memory area at the pointer. Otherwise, this can lead to a system crash or invalid data.

**Required License**

TC3 Vision Base
6.4.4.16  F_VN_EraseFromContainer

Erase elements from a container.

Syntax

Definition:

FUNCTION F_VN_EraseFromContainer : HRESULT
VAR_INPUT
    ipContainer    : ITCvVnContainer;
    nStartIdx      : ULLINT;
    nCount         : ULLINT;
    hrPrev         : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvVnContainer</td>
<td>Container from which to erase elements</td>
</tr>
<tr>
<td>nStartIdx</td>
<td>ULLINT</td>
<td>Index of the first element to erase</td>
</tr>
<tr>
<td>nCount</td>
<td>ULLINT</td>
<td>Number of elements to erase</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
**F_VN_ExportContainer**

Export the container elements into a buffer (e.g. an array). Only possible for containers with basic elements.

**Syntax**

**Definition:**

```
FUNCTION F_VN_ExportContainer : HRESULT
VAR_INPUT
   ipContainer : ITCvNContainer;
   pBuffer     : PVOID;
   nBufferSize : ULINT;
   hrPrev      : HRESULT;
END_VAR
```  

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>pBuffer</td>
<td>PVOID</td>
<td>Buffer to store the container elements (Make sure to allocate enough memory! The required size in bytes can be determined using the function F_VN_ExportContainerSize)</td>
</tr>
<tr>
<td>nBufferSize</td>
<td>ULINT</td>
<td>Size of the buffer memory in bytes</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Further information**

The function **F_VN_ExportContainer** exports a container to a data buffer. Make sure that sufficient memory is allocated at the place of the pointer in order to write the container data to it! The required size in bytes can be determined with the function **F_VN_ExportContainerSize** [698]. You can allocate the memory, for example, by creating an array of the appropriate size.

**Application**

The export of a container of the type **Vector_REAL** with 10 elements looks like this, for example:

```
VAR
   aArray: ARRAY [0..9] OF REAL;
   ipContainer: ITCvNContainer;
   nBufferSize: ULINT;
END_VAR
hr := F_VN_ExportContainerSize(ipContainer, nBufferSize, hr);
IF nBufferSize = SIZEOF(aArray) THEN
   hr := F_VN_ExportContainer(  
      ipContainer:=ipContainer,
      pBuffer:=ADR(aArray),
      nBufferSize:=nBufferSize,
      hr
   );
END_IF
```
## Memory access

It must be ensured that the type and length of the container correspond to the size of the memory area at the pointer. Otherwise, this can lead to a system crash or invalid data.

### Related functions

- F_VN_ExportContainer \([\text{696}]\)
- F_VN_ExportSubContainer \([\text{699}]\)
- F_VN_ExportContainer_String \([\text{697}]\)
- F_VN_ExportSubContainer_String \([\text{700}]\)
- F_VN_ExportContainerSize \([\text{698}]\)
- F_VN_ExportSubContainerSize \([\text{702}]\)

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.18 F_VN_ExportContainer_String

Function `F_VN_ExportContainer_String` takes

- `ipContainer` : `ITcVnContainer`
- `sText` : `STRING`
- `nMaxLength` : `ULINT`
- `hrPrev` : `HRESULT`

Export the container elements into a string. Only possible for containers of type `ContainerType_String_SINT`.

### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_ExportContainer_String : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    sText : STRING;
    nMaxLength : ULINT;
    hrPrev : HRESULT;
END_VAR
```
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container of type ContainerType_String_SINT</td>
</tr>
<tr>
<td>sText</td>
<td>STRING</td>
<td>Make sure to choose a sufficient STRING size! The required size can be determined using the function VnExportContainerSize.</td>
</tr>
<tr>
<td>nMaxLength</td>
<td>ULINT</td>
<td>Maximum string length to export (including 0 termination). If the container content is longer, the string is cut off at nMaxLength - 1 and 0 termination is appended. In this case, S_FALSE is returned.</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Related functions

- F_VN_ExportContainer [696]
- F_VN_ExportSubContainer [699]
- F_VN_ExportContainer_String [697]
- F_VN_ExportSubContainer_String [700]
- F_VN_ExportContainerSize [698]
- F_VN_ExportSubContainerSize [702]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.19 F_VN_ExportContainerSize

Determine the required buffer size in bytes to store all container elements (number_of_Elements * size_per_Element). Only possible for containers with basic elements.

Syntax

Definition:

FUNCTION F_VN_ExportContainerSize : HRESULT 
VAR_INPUT
  _ipContainer : ITcVnContainer;
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>nBufferSize</td>
<td>Reference To ULINT</td>
<td>Output parameter containing the required buffer size</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Related functions**

- F_VN_ExportContainer [696]
- F_VN_ExportSubContainer [699]
- F_VN_ExportContainer_String [697]
- F_VN_ExportSubContainer_String [700]
- F_VN_ExportContainerSize [698]
- F_VN_ExportSubContainerSize [702]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.20 F_VN_ExportSubContainer

Export the container elements of a sub-container into a buffer (e.g. an array). Only possible for 2-dimensional containers with basic elements.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_ExportSubContainer : HRESULT
VAR_INPUT
    ipContainer : ITCvNContainer;
    nIndex      : ULINT;
    pBuffer     : PVOID;
    nBufferSize : Reference To ULINT;
    hrPrev      : HRESULT;
END_VAR
```
**API reference**

nBufferSize : ULINT;
hrPrev : HRESULT;
END_VAR

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>pBuffer</td>
<td>PVOID</td>
<td>Buffer to store the container elements (Make sure to allocate enough memory! The required size in bytes can be determined using the function F_VN_ExportSubContainerSize)</td>
</tr>
<tr>
<td>nBufferSize</td>
<td>ULINT</td>
<td>Size of the buffer memory in bytes</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

**Related functions**

- F_VN_ExportContainer [696]
- F_VN_ExportSubContainer [699]
- F_VN_ExportContainer_String [697]
- F_VN_ExportSubContainer_String [700]
- F_VN_ExportContainerSize [698]
- F_VN_ExportSubContainerSize [702]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.4.21 F_VN_ExportSubContainer_String**

Export the container elements of a sub-container into a string. Only possible for 2-dimensional containers of type ContainerType_Vector_String_SINT.

**Syntax**

Definition:
FUNCTION F_VN_ExportSubContainer_String : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    sText       : STRING;
    nMaxLength  : ULINT;
    hrPrev      : HRESULT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container of type ContainerType_Vector_String_SINT</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>sText</td>
<td>STRING</td>
<td>Make sure to choose a sufficient STRING size! The required size can be determined using the function VnExportSubContainerSize.</td>
</tr>
<tr>
<td>nMaxLength</td>
<td>ULINT</td>
<td>Maximum string length to export (including 0 termination). If the container content is longer, the string is cut off at nMaxLength - 1 and 0 termination is appended. In this case, S_FALSE is returned.</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Application**

The function F_VN_ExportSubContainer is used, for example, to obtain the result text in the case of Code Reading [753]:

```plaintext
hr := F_VN_ExportSubContainer_String(ipDecodedText, 0, sCodeAsString, 255, hr);
// sCodeAsString contains the result
```

**Related functions**

- F_VN_ExportContainer [696]
- F_VN_ExportSubContainer [699]
- F_VN_ExportContainer_String [697]
- F_VN_ExportSubContainer_String [700]
- F_VN_ExportContainerSize [698]
- F_VN_ExportSubContainerSize [702]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
Determine the required buffer size in bytes to store all container elements of a sub-container (number_of_Elements * size_per_Element). Only possible for 2-dimensional containers with basic elements.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_ExportSubContainerSize : HRESULT
VAR_INPUT
  ipContainer : ITCvNContainer;
  nIndex : ULINT;
  nBufferSize : Reference To ULINT;
  hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer</td>
<td>Container with basic elements</td>
</tr>
<tr>
<td>nIndex</td>
<td>ULINT</td>
<td>Index of the requested element</td>
</tr>
<tr>
<td>nBufferSize</td>
<td>Reference To ULINT</td>
<td>Output parameter containing the required buffer size</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Related functions**

- F_VN_ExportContainer [696]
- F_VN_ExportSubContainer [699]
- F_VN_ExportContainer_String [697]
- F_VN_ExportSubContainer_String [700]
- F_VN_ExportContainerSize [698]
- F_VN_ExportSubContainerSize [702]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.4.23  F_VN_ExtractContainerRange

Copy the specified range of the source container into the destination container. (If the destination container already contains elements, they will be removed.)

Syntax

Definition:

```c
FUNCTION F_VN_ExtractContainerRange : HRESULT
VAR_INPUT
    ipSrcContainer : ITCvnContainer;
    ipDestContainer : Reference To ITCvnContainer;
    nFirstIdx : ULINT;
    nLastIdx : ULINT;
    hrPrev : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcContainer</td>
<td>ITCvnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>ipDestContainer</td>
<td>Reference To ITCvnContainer</td>
<td>Destination container</td>
</tr>
<tr>
<td>nFirstIdx</td>
<td>ULINT</td>
<td>Index of the first element to copy</td>
</tr>
<tr>
<td>nLastIdx</td>
<td>ULINT</td>
<td>Index of the last element to copy</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.24  F_VN_GetContainer
Gets a pointer to the current element converted into an ITcVnContainer interface and increment its reference counter (only possible for container types). (Alternatively use interface method .GetContainer.)

**Syntax**

**Definition:**

```c
FUNCTION F_VN_GetContainer : HRESULT
VAR_INPUT
  ipIterator : ITcVnForwardIterator;
  ipContainer : Reference To ITcVnContainer;
  hrPrev      : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipIterator</td>
<td>ITcVnForwardIterator</td>
<td>Iterator</td>
</tr>
<tr>
<td>ipContainer</td>
<td>Reference To ITcVnContainer</td>
<td>Returns the container interface</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Further information**

**Deep copy**

This function creates a deep copy of the container element. This means that if you wish to change the element in the container, you must write the changes back to the container separately.

If you wish to increment the iterator at the same time as accessing the container element, the expert version F_VN_GetContainerExp [704] is available for this.

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.4.25 F_VN_GetContainerExp**

```c
F_VN_GetContainerExp
VAR_INPUT
  ipIterator : ITcVnForwardIterator;
  ipContainer : Reference To ITcVnContainer;
  nOffset : DINT;
  hrPrev : HRESULT;
END_VAR
```
Gets a pointer to the current element converted into an ITcVnContainer interface and increment its reference counter (only possible for container types). Additionally, an offset to increment or decrement the iterator afterwards can be provided. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_GetContainerExp : HRESULT
VAR_INPUT
  ipIterator : ITcVnForwardIterator;
  ipContainer : Reference To ITcVnContainer;
  nOffset     : DINT;
  hrPrev      : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipIterator</td>
<td>ITcVnForwardIterator</td>
<td>Iterator</td>
</tr>
<tr>
<td>ipContainer</td>
<td>Reference To ITcVnContainer</td>
<td>Returns the container interface</td>
</tr>
<tr>
<td>nOffset</td>
<td>DINT</td>
<td>Offset to increment (&gt;0) or decrement (&lt;0) the iterator afterwards</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Further information

The function F_VN_GetContainerExp is the expert version of F_VN_GetContainer [703]. It helps you if, at the same time as accessing a container element by means of an iterator, you also wish to increment this iterator. Set the parameter nOffset to the value 1 for a normal, sequential incrementation:

```
hr := F_VN_GetContainerExp(ipIterator, ipElement, 1, hr);
```

With the standard function you would need two function calls for this:

```
hr := F_VN_GetContainer(ipIterator, ipElement, hr);
hr := F_VN_IncrementIterator(ipIterator, hr);
```

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
### 6.4.4.26 F_VN_GetForwardIterator

**Syntax**

Definition:

```plaintext
FUNCTION F_VN_GetForwardIterator : HRESULT
VAR_INPUT
  ipContainer : ITcVnContainer;
  ipIterator  : Reference To ITcVnForwardIterator;
  hrPrev      : HRESULT;
END_VAR
```

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container</td>
</tr>
<tr>
<td>ipIterator</td>
<td>Reference To ITcVnForwardIterator</td>
<td>Returns the iterator interface</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

#### Return value

HRESULT (135)

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.27 F_VN_GetNumberOfElements

**Syntax**

Definition:

```plaintext
FUNCTION F_VN_GetNumberOfElements : HRESULT
VAR_INPUT
  ipContainer : ITcVnContainer;
  nNumberOfElements : Reference To ULINT;
  hrPrev      : HRESULT;
END_VAR
```

Gets the number of elements in the container. (Alternatively use interface method .getElementNum.)

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
FUNCTION F_VN_GetNumberOfElements : HRESULT
VAR_INPUT
    ipContainer : ITCvNContainer;
    nNumberOfElements : Reference To ULINT;
    hrPrev : HRESULT;
END_VAR

.Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer [345]</td>
<td>Container</td>
</tr>
<tr>
<td>nNumberOfElements</td>
<td>Reference To ULINT</td>
<td>Returns the number of elements in ipContainer</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

.Return value

HRESULT [135]

.Further information

This function should not be used to iterate over the elements of a container. Use iterators to access the elements [144] instead. In particular, the following code construct is not recommended:

```pascal
hr := F_VN_GetNumberOfElements(ipContainer, ipNumber, hr);
FOR i:=0 TO (ipNumber-1) DO
    // access container elements
END_FOR
```

Since `ipNumber` is a UDINT (natural integer), a return value of `ipNumber = 0` would cause an underflow.

.Required License

TC3 Vision Base

.System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.28 F_VN_GetRandomAccessIterator

Gets a random access iterator for the container. (Alternatively use interface method .GetRandomAccessIterator.)

.Syntax

Definition:

FUNCTION F_VN_GetRandomAccessIterator : HRESULT
VAR_INPUT
    ipContainer : ITCvNContainer;
    ipIterator : Reference To ITCvNRandomAccessIterator;
    hrPrev : HRESULT;
END_VAR
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container</td>
</tr>
<tr>
<td>ipIterator</td>
<td>Reference To ITcVnRandomAccessIterator [342]</td>
<td>Returns the iterator interface</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.29 F_VN_IncrementIterator

```
FUNCTION F_VN_IncrementIterator : HRESULT
VAR_INPUT
    ipIterator : ITcVnForwardIterator;
    hrPrev : HRESULT;
END_VAR
```

Increment the iterator. (Alternatively use interface method .Increment.)

Syntax

Definition:

```
FUNCTION F_VN_IncrementIterator : HRESULT
VAR_INPUT
    ipIterator : ITcVnForwardIterator;
    hrPrev : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipIterator</td>
<td>ITcVnForwardIterator [336]</td>
<td>Iterator</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]
Further information

The function `F_VN_IncrementIterator` increments an iterator [334] by one element. If instead of that you wish to decrement the iterator or to increment it by several positions, use the function `F_VN_AdvanceIterator` [685].

Parameter

Iterator

The parameter `ipIterator` is of the type `ITcVnForwardIterator`. Since `ITcVnBidirectionalIterator` and `ITcVnRandomAccessIterator` inherit from it, these can also be transferred to the function. If a forward iterator is to be decremented, an internal attempt is made to transform it into a bidirectional iterator and to decrement it.

Related functions

- `F_VN_IncrementIterator` [708]
- `F_VN_AdvanceIterator` [685]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.30  F_VN_IteratorDistance

Computes the distance between 2 iterators.

Syntax

Definition:

```c
FUNCTION F_VN_IteratorDistance : HRESULT
VAR_INPUT
    ipIteratorFirst : ITcVnForwardIterator;
    ipIteratorLast : ITcVnForwardIterator;
    nDistance : Reference To LINT;
    hrPrev : HRESULT;
END_VAR
```

The function `F_VN_IncrementIterator` increments an iterator by one element. If instead of that you wish to decrement the iterator or to increment it by several positions, use the function `F_VN_AdvanceIterator`.
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipIteratorFirst</td>
<td>ITcVnForwardIterator [336]</td>
<td>First iterator</td>
</tr>
<tr>
<td>ipIteratorLast</td>
<td>ITcVnForwardIterator [336]</td>
<td>Last iterator</td>
</tr>
<tr>
<td>nDistance</td>
<td>Reference To LINT</td>
<td>Returns the computed distance</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.4.31 F_VN_ ReserveContainerMemory

Reserve container memory (call with maximum required number of elements before manually appending elements for better performance)

### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_ReserveContainerMemory : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nElements : ULINT;
    hrPrev : HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container for which to reserve the memory</td>
</tr>
<tr>
<td>nElements</td>
<td>ULINT</td>
<td>Number of elements for which the container should reserve memory</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.4.32 F_VN_ReverseContainer

Reverse the container elements.

**Syntax**

**Definition:**

```
FUNCTION F_VN_ReverseContainer : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Container to be reversed</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.4.33  F_VN_SetContainer

Sets the current element using an ITcVnContainer interface (only possible for container types). (Alternatively use interface method .SetContainer.)

Syntax

Definition:

FUNCTION F_VN_SetContainer : HRESULT
VAR_INPUT
  ipIterator : ITcVnForwardIterator;
  ipContainer : ITcVnContainer;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipIterator</td>
<td>ITcVnForwardIterator</td>
<td>Iterator</td>
</tr>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Container interface of which the content is to be assigned to the current element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.4.34  F_VN_SetIteratorToBegin

Sets the iterator to the first element of the container. (Alternatively use interface method .SetToBegin.)

Syntax

Definition:
FUNCTION F_VN_SetIteratorToBegin : HRESULT
VAR_INPUT
    ipIterator : ITcVnForwardIterator;
    hrPrev      : HRESULT;
END_VAR

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipIterator</td>
<td>ITcVnForwardIterator [336]</td>
<td>Iterator</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.5 Basic Image Operations

The group contains functions for handling images [140].

#### Functions

**Creation of images**

- F_VN_ConvertElementType(Exp) [715]
- F_VN_CreateAssociatedImage [722]
- F_VN_CreateEmptyImage [723]
- F_VN_CreateImage [724]
- F_VN_CreateImageAndSetPixels [725]
- F_VN_CreateImageFromArray [726]

**Image information**

- F_VN_GetImageHeight [736]
- F_VN_GetImageInfo [737]
- F_VN_GetImageWidth [738]
- F_VN_GetPixelFormat [739]

**Copying images**

- F_VN_CopyImage [717]
- F_VN_CopyImageRegion [718]
- F_VN_CopyImageRegionToRegion [720]
Pixel manipulation

- F_VN_GetPixel [739]
- F_VN_SetPixel [744]
- F_VN_SetPixels(Exp) [745]

Region of Interest

- F_VN_GetRoi [740]
- F_VN_ResetRoi [742]
- F_VN_SetRoi [747]
- F_VN_SetRoi_TcVnRectangle_UDINT [748]

Exporting image content to memory area

- F_VN_ExportImage [727]
- F_VN_ExportImageAsBmp(Exp) [728]
- F_VN_ExportImageSize [731]
- F_VN_ExportImageAsBmpSize(Exp) [730]

Fusing images

- F_VN_FuseImages [732]
- F_VN_FuseImagesArray [734]

Handling image channels

- F_VN_CombineImageChannels [714]
- F_VN_GetImageChannel [735]
- F_VN_MixImageChannels [741]
- F_VN_SetImageChannel [743]
- F_VN_SplitImageChannels [749]

Displaying images

- F_VN_CopyIntoDisplayableImage [721]
- F_VN_TransformIntoDisplayableImage(Exp) [750]

Samples

Basic Image Operations [1366]

6.4.5.1 F_VN_CombineImageChannels

```
F_VN_CombineImageChannels

pSrcImages PVOID HRESULT F_VN_CombineImageChannels
ipDestImage Reference To ITcVnImage
nSrcArraySize UINT
hrPrev HRESULT
```

Combines several single-channel images into one multi-channel image.

Syntax

Definition:
FUNCTION F_VN_CombineImageChannels : HRESULT
VAR_INPUT
  pSrcImages : PVOID;
  ipDestImage : Reference To ITcVnImage;
  nSrcArraySize : UINT;
  hrPrev : HRESULT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pSrcImages</td>
<td>PVOID</td>
<td>Pointer to an array of single-channel source images</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage[383]</td>
<td>Multi-channel destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>nSrcArraySize</td>
<td>UINT</td>
<td>pSrcImages array size</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.5.2 F_VN_ConvertElementType

Converts an image to another element type.

**Syntax**

**Definition:**

FUNCTION F_VN_ConvertElementType : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  eElementType : ETcVnElementType;
  hrPrev : HRESULT;
END_VAR

Converts an image to another element type.
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage ![383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage ![383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>eElementType</td>
<td>ETCvNElementType ![186]</td>
<td>Desired element type of the destination image</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT ![135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT ![135]

**Required License**
TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.5.3 F_VN_ConvertElementTypeExp

Converts an image to another element type. (expert function)

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_ConvertElementTypeExp : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    eElementType : ETCvNElementType;
    fScaleFactor : LREAL;
    fDelta : LREAL;
    hrPrev : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>eElementType</td>
<td>ETcVnElementType [186]</td>
<td>Desired element type of the destination image</td>
</tr>
<tr>
<td>fScaleFactor</td>
<td>LREAL</td>
<td>Scale factor for the pixel values</td>
</tr>
<tr>
<td>fDelta</td>
<td>LREAL</td>
<td>Value that is added to the scaled pixel values</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.5.4 F_VN_CopyImage

Creates a deep copy of an image.

#### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_CopyImage : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev : HRESULT;
END_VAR
```

## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Further information

The function `F_VN_CopyImage` creates a deep copy of the input image `ipSrcImage`. Alternatively, the function `F_VN_CopyImageRegion` can be used to create a copy of an image region.

Computing time optimization

With regard to computing time optimization, check each image copy for its necessity.

Related functions

- `F_VN_CopyImage` for copying the entire image
- `F_VN_CopyImageRegion` for copying an image region
- `F_VN_CopyImageRegionToRegion` for copying an image region to a certain position

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g.</td>
<td>Tc3_Vision</td>
</tr>
<tr>
<td></td>
<td>Intel 4-core Atom CPU</td>
<td></td>
</tr>
</tbody>
</table>

6.4.5.5 **F_VN_CopyImageRegion**

Deep copy the specified region of interest into a new image.

Syntax

Definition:

```c
FUNCTION F_VN_CopyImageRegion : HRESULT
VAR_INPUT
  ipSrcImage : ITCvImage;
  nX          : UDINT;
  nY          : UDINT;
  nWidth      : UDINT;
  nHeight     : UDINT;
  ipDestImage : Reference To ITCvImage;
  hrPrev      : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>nX</td>
<td>UDINT</td>
<td>Left boundary (inclusive 0-based index)</td>
</tr>
<tr>
<td>nY</td>
<td>UDINT</td>
<td>Upper boundary (inclusive 0-based index)</td>
</tr>
<tr>
<td>nWidth</td>
<td>UDINT</td>
<td>ROI width</td>
</tr>
<tr>
<td>nHeight</td>
<td>UDINT</td>
<td>ROI height</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Further information

The function **F_VN_CopyImageRegion** creates a deep copy of an image region of the input image `ipSrcImage`. The image region is specified by a rectangle with position and size, as with an ROI [149].

#### Computing time optimization

With regard to computing time optimization, check each image copy for its necessity.

If a copy is not required, but instead only the image region is to be reduced in size, a Region of Interest [149] can be set instead.

### Application

The copying of an image region of the size [240, 120] with the top left corner at the position [50, 50] looks like this, for example:

```c
hr := F_VN_CopyImageRegion(
    ipSrcImage := ipImageIn,
    nX := 50,
    nY := 50,
    nWidth := 240,
    nHeight := 120,
    ipDestImage := ipImageWork,
    hrPrev := hr,
);
```

### Related functions

- **F_VN_CopyImage** [717] for copying the entire image
- **F_VN_CopyImageRegion** [718] for copying an image region
- **F_VN_CopyImageRegionToRegion** [720] for copying an image region to a certain position

### Required License

TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.5.6 **F_VN_CopyImageRegionToRegion**

**F_VN_CopyImageRegionToRegion**

Copy an image region into another image region.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_CopyImageRegionToRegion : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    nXSrc : UDINT;
    nYSrc : UDINT;
    nWidth : UDINT;
    nHeight : UDINT;
    ipDestImage : ITcVnImage;
    nXDest : UDINT;
    nYDest : UDINT;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image</td>
</tr>
<tr>
<td>nXSrc</td>
<td>UDINT</td>
<td>Left boundary in ipSrcImage (inclusive 0-based index)</td>
</tr>
<tr>
<td>nYSrc</td>
<td>UDINT</td>
<td>Upper boundary in ipSrcImage (inclusive 0-based index)</td>
</tr>
<tr>
<td>nWidth</td>
<td>UDINT</td>
<td>ROI width</td>
</tr>
<tr>
<td>nHeight</td>
<td>UDINT</td>
<td>ROI height</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITcVnImage</td>
<td>Destination image (same type as ipSrcImage)</td>
</tr>
<tr>
<td>nXDest</td>
<td>UDINT</td>
<td>Left boundary in ipDestImage (inclusive 0-based index)</td>
</tr>
<tr>
<td>nYDest</td>
<td>UDINT</td>
<td>Upper boundary in ipDestImage (inclusive 0-based index)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

**HRESULT [135]**
Further information

The function `F_VN_CopyImageRegionToRegion` copies an image region of the input image `ipSrcImage` to a place in the destination image `ipDestImage`. The image region is specified by a rectangle with position and size, as with an ROI [149].

Application

The copying of an image area with a size `[100, 100]` at the position `aPosition` in the top left corner of the result image `ipImageRes` looks like this, for example:

```cpp
hr := F_VN_CopyImageRegionToRegion(
    ipSrcImage := ipImageWork,
    nXSrc := aPosition[0],
    nYSrc := aPosition[1],
    nWidth := 100,
    nHeight := 100,
    ipDestImage := ipImageRes,
    nXDest := 0,
    nYDest := 0,
    hrPrev := hr
)
```

Samples

- Copy image areas [1367]

Related functions

- `F_VN_CopyImage` [717] for copying the entire image
- `F_VN_CopyImageRegion` [718] for copying an image region
- `F_VN_CopyImageRegionToRegion` [720] for copying an image region to a certain position

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.5.7 `F_VN_CopyIntoDisplayableImage`

Copys an image into a displayable image. If you do not want to use `ipSrcImage` after this function call, you might want to use `F_VN_TransformIntoDisplayableImage` instead for better performance.

Syntax

Definition:

```cpp
FUNCTION F_VN_CopyIntoDisplayableImage : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnDisplayableImage;
    hrPrev := HRESULT;
END_VAR
```

Copys an image into a displayable image. If you do not want to use `ipSrcImage` after this function call, you might want to use `F_VN_TransformIntoDisplayableImage` instead for better performance.
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnDisplayableImage [383]</td>
<td>Returns the displayable image</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Further information

For further details, see Displayable images [142].

## Related functions

- F_VN_TransformIntoDisplayableImage [750]
- F_VN_TransformIntoDisplayableImageExp [751]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.5.8 F_VN_CreateAssociatedImage

F_VN_CreateAssociatedImage

ipSrcImage ITcVnImage

ipDestImage Reference To ITcVnImage

hrPrev HRESULT

F_VN_CreateAssociatedImage HRESULT

Creates a new image that shares its data with the source image. E.g. useful to work on different (disjoint) ROIs in parallel.

## Syntax

**Definition:**

FUNCTION F_VN_CreateAssociatedImage : HRESULT

VAR_INPUT

  ipSrcImage : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  hrPrev : HRESULT;

END_VAR
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage[383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage[383]</td>
<td>Returns the created image (Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.5.9  F_VN_CreateEmptyImage

**Definition:**

FUNCTION F_VN_CreateEmptyImage : HRESULT
VAR_INPUT
    ipImage : Reference To ITcVnImage;
    hrPrev : HRESULT;
END_VAR

Creates an empty image without allocating any data buffer. The initial reference count is set to one if a new image interface is created and kept, otherwise.

**Syntax**

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage</td>
<td>Reference To ITcVnImage[383]</td>
<td>Returns the created image</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]
6.4.5.10 **F_VN_CreateImage**

Creates an image and allocate an appropriate data buffer. The initial reference count is set to one if a new image interface is created and kept, otherwise.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_CreateImage : HRESULT
VAR_INPUT
    ipImage : Reference To ITCvNImage;
    nWidth : UDINT;
    nHeight : UDINT;
    ePixelType : ETCvNElementType;
    nChannelNum : UDINT;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage</td>
<td>Reference To ITCvNImage</td>
<td>Returns the created image (Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>nWidth</td>
<td>UDINT</td>
<td>Image width</td>
</tr>
<tr>
<td>nHeight</td>
<td>UDINT</td>
<td>Image height</td>
</tr>
<tr>
<td>ePixelType</td>
<td>ETCvNElementType</td>
<td>Pixel type</td>
</tr>
<tr>
<td>nChannelNum</td>
<td>UDINT</td>
<td>Number of channels (1 to 255)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base
6.4.5.11  **F_VN_CreateImageAndSetPixels**

Creates an image, allocates an appropriate data buffer and sets all pixels to the specified value. The initial reference count is set to one if a new image interface is created and kept, otherwise.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_CreateImageAndSetPixels : HRESULT
VAR_INPUT
  ipImage : Reference To ITCnVnImage;
  nWidth : UDINT;
  nHeight : UDINT;
  ePixelType : ETCnVnElementType;
  nChannelNum : UDINT;
  aValue : Reference To TcVnVector4_LREAL;
  hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage</td>
<td>Reference To ITCnVnImage</td>
<td>Returns the created image (Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>nWidth</td>
<td>UDINT</td>
<td>Image width</td>
</tr>
<tr>
<td>nHeight</td>
<td>UDINT</td>
<td>Image height</td>
</tr>
<tr>
<td>ePixelType</td>
<td>ETCnVnElementType</td>
<td>Pixel type</td>
</tr>
<tr>
<td>nChannelNum</td>
<td>UDINT</td>
<td>Number of channels (1 to 4)</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector4_REAL</td>
<td>Pixel value</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.5.12 F_VN_CreateImageFromArray

**F_VN_CreateImageFromArray**

```c
FUNCTION F_VN_CreateImageFromArray : HRESULT
VAR_INPUT
  pData : PVOID;
  ipImage : Reference To ITcVnImage;
  nWidth : UDINT;
  nHeight : UDINT;
  ePixelType : ETcVnElementType;
  nChannelNum : UDINT;
  hrPrev : HRESULT;
END_VAR
```

Creates an image and initialize it with the provided data. The initial reference count is set to one if a new image interface is created and kept, otherwise.

**Syntax**

**Definition:**

FUNCTION F_VN_CreateImageFromArray : HRESULT
VAR_INPUT
  pData : PVOID;
  ipImage : Reference To ITcVnImage;
  nWidth : UDINT;
  nHeight : UDINT;
  ePixelType : ETcVnElementType;
  nChannelNum : UDINT;
  hrPrev : HRESULT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pData</td>
<td>PVOID</td>
<td>Pointer to the 1D data array. Make sure that the array contains at least nWidth * nHeight * nChannelNum elements and that the array element type matches ePixelType.</td>
</tr>
<tr>
<td>ipImage</td>
<td>Reference To ITcVnImage</td>
<td>Returns the created image (Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>nWidth</td>
<td>UDINT</td>
<td>Image width</td>
</tr>
<tr>
<td>nHeight</td>
<td>UDINT</td>
<td>Image height</td>
</tr>
<tr>
<td>ePixelType</td>
<td>ETcVnElementType</td>
<td>Pixel type</td>
</tr>
<tr>
<td>nChannelNum</td>
<td>UDINT</td>
<td>Number of channels (1 to 255)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [383]

**Required License**

TC3 Vision Base
### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

#### 6.4.5.13 F_VN_ExportImage

**Syntax**

**Definition:**

```c
FUNCTION F_VN_ExportImage : HRESULT
VAR_INPUT
    ipImage : ITCvNImage;
    pBuffer : PVOID;
    nBufferSize : ULINT;
    hrPrev : HRESULT;
END_VAR
```

Exports the image data into a given buffer (e.g. an array). F_VN_ExportImageSize should be called before to get the required buffer size.

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage</td>
<td>ITCvNImage</td>
<td>Image</td>
</tr>
<tr>
<td>pBuffer</td>
<td>PVOID</td>
<td>Pointer to a buffer to store the image data (Make sure to allocate enough memory! The required size in bytes can be determined using the function F_VN_ExportImageSize)</td>
</tr>
<tr>
<td>nBufferSize</td>
<td>ULINT</td>
<td>Size of the buffer memory in bytes</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

#### Return value

**HRESULT [135]**

#### Required License

TC3 Vision Base

#### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

---

**BECKHOFF**

API reference

TF7000 - TF7300 Version: 1.3 727
6.4.5.14  F_VN_ExportImageAsBmp

Exports the image as bitmap into a given buffer (e.g. an array). F_VN_ExportImageAsBmpSize should be called before to get the required buffer size. Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

FUNCTION F_VN_ExportImageAsBmp : HRESULT
VAR_INPUT
    ipImage : ITCvNImage;
    pBuffer : PVOID;
    nBufferSize : ULINT;
    hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage</td>
<td>ITCvNImage</td>
<td>Image</td>
</tr>
<tr>
<td>pBuffer</td>
<td>PVOID</td>
<td>Pointer to a buffer to store the bitmap image (Make sure to allocate enough memory! The required size in bytes can be determined using the function F_VN_ExportImageAsBmpSize)</td>
</tr>
<tr>
<td>nBufferSize</td>
<td>ULINT</td>
<td>Size of the buffer memory in bytes</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Further information

Note the conversion properties of the interface ITCvNBitmapExport [371].

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.5.15  F_VN_ExportImageAsBmpExp

Exports the image as bitmap into a given buffer (e.g. an array). F_VN_ExportImageAsBmpSizeExp should be called before to get the required buffer size. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:
FUNCTION F_VN_ExportImageAsBmpExp : HRESULT
VAR_INPUT
    ipImage : ITcVnImage;
    pBuffer : PVOID;
    nBufferSize : ULngINT;
    nWidth : ULngINT;
    nHeight : ULngINT;
    hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage</td>
<td>ITcVnImage</td>
<td>Image</td>
</tr>
<tr>
<td>pBuffer</td>
<td>PVOID</td>
<td>Pointer to a buffer to store the bitmap image (Make sure to allocate enough memory! The required size in bytes can be determined using the function F_VN_ExportImageAsBmpSizeExp)</td>
</tr>
<tr>
<td>nBufferSize</td>
<td>ULngINT</td>
<td>Size of the buffer memory in bytes</td>
</tr>
<tr>
<td>nWidth</td>
<td>ULngINT</td>
<td>Desired width (or 0 to keep the original width)</td>
</tr>
<tr>
<td>nHeight</td>
<td>ULngINT</td>
<td>Desired height (or 0 to keep the original height)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.5.16  F_VN_ExportImageAsBmpSize

Get the required buffer size (in bytes) to store the image as bitmap.

Syntax

Definition:
FUNCTION F_VN_ExportImageAsBmpSize : HRESULT
VAR_INPUT
  ipImage   : ITcVnImage;
  nBufferSize : Reference To ULINT;
  hrPrev    : HRESULT;
END_VAR

Inputs

Name         Type                Description
------------ ---------------------- ----------------------------------
ipImage      ITcVnImage [383]    Image
nBufferSize  Reference To ULINT Returns the required buffer size in bytes
hrPrev       HRESULT [135]      HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.5.17  F_VN_ExportImageAsBmpSizeExp

Get the required buffer size (in bytes) to store the image as bitmap. (expert function)

Syntax

Definition:
FUNCTION F_VN_ExportImageAsBmpSizeExp : HRESULT

VAR_INPUT
    ipImage : ITcVnImage;
    nBufferSize : Reference To ULINT;
    nWidth : UDINT;
    nHeight : UDINT;
    hrPrev : HRESULT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage</td>
<td>ITcVnImage [383]</td>
<td>Image</td>
</tr>
<tr>
<td>nBufferSize</td>
<td>Reference To ULINT</td>
<td>Returns the required buffer size in bytes</td>
</tr>
<tr>
<td>nWidth</td>
<td>UDINT</td>
<td>Desired width (or 0 to keep the original width)</td>
</tr>
<tr>
<td>nHeight</td>
<td>UDINT</td>
<td>Desired height (or 0 to keep the original height)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.5.18 **F_VN_ExportImageSize**

Get the required buffer size (in bytes) to store the image data.

**Syntax**

Definition:

FUNCTION F_VN_ExportImageSize : HRESULT

VAR_INPUT
    ipImage : ITcVnImage;
    nBufferSize : Reference To ULINT;
    hrPrev : HRESULT;
END_VAR
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage</td>
<td>ITcVnImage [383]</td>
<td>Image</td>
</tr>
<tr>
<td>nBufferSize</td>
<td>Reference To ULINT</td>
<td>Returns the required buffer size in bytes</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.5.19 **F_VN_FuseImages**

Fuse 2 images vertically (intended for line scan cameras).

**Syntax**

**Definition:**

FUNCTION F_VN_FuseImages : HRESULT
VAR_INPUT
  ipSrcImage1 : ITcVnImage;
  ipSrcImage2 : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  nFirstLine : UDINT;
  nNumLines : UDINT;
  hrPrev : HRESULT;
END_VAR
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage1</td>
<td>ITcVnImage [383]</td>
<td>Source image 1</td>
</tr>
<tr>
<td>ipSrcImage2</td>
<td>ITcVnImage [383]</td>
<td>Source image 2</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image</td>
</tr>
<tr>
<td>nFirstLine</td>
<td>UDINT</td>
<td>Line index in ipSrcImage1, which is the first line in ipDestImage</td>
</tr>
<tr>
<td>nNumLines</td>
<td>UDINT</td>
<td>The number of lines that should be copied to ipDestImage, starting with nFirstLine in ipSrcImage1. Once the last line of ipSrcImage1 was copied, the remaining lines are copied from ipSrcImage2, starting at line index 0.</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Further information

The function `F_VN_FuseImages` joins two images to form a new contiguous image. This is of particular interest for line scan cameras.

The function `F_VN_FuseImagesArray [734]` can be used to fuse up to 10 images.

### Application

Line scan cameras usually return a given number of successively captured lines as a 2D image. This 2D image can contain an entire object of interest (shown in green in the figure below), or it can contain part of an object (shown in red in the figure).
To analyze the red object, the function `F_VN_FuseImages` can be used to fuse the two object parts to form an complete image. If it is known which lines relate to the object, the linking can be limited to the relevant lines.

```c
F_VN_FuseImages(ipImage1, ipImage2, ipFusedImage, 6, 7, S_OK);
```

### Samples
- Fusing several images [1366]

### Related functions
- `F_VN_FuseImages` [732] for fusing two images
- `F_VN_FuseImagesArray` [734] for fusing up to 10 images

### Required License
TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.5.20 `F_VN_FuseImagesArray`

Fusion up to 10 images vertically (intended for line scan cameras).

### Syntax

**Definition:**

```c
FUNCTION F_VN_FuseImagesArray : HRESULT
VAR_INPUT
    aSrcImageArray : Reference To TcVnArray10_ITcVnImage;
    nValidImages   : UDINT;
    ipDestImage    : Reference To ITcVnImage;
    nFirstLine     : UDINT;
    nNumLines      : UDINT;
    hrPrev          : HRESULT;
END_VAR
```

```c
F_VN_FuseImagesArray
```

Fuse up to 10 images vertically (intended for line scan cameras).
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aSrcImageArray</td>
<td>Reference To TcVnArray10 ITcVnImage</td>
<td>Ten-element source image array (not all elements need to be filled, nValidImages specifies the actual amount of images)</td>
</tr>
<tr>
<td>nValidImages</td>
<td>UDINT</td>
<td>Number of valid images in aSrcImageArray</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image</td>
</tr>
<tr>
<td>nFirstLine</td>
<td>UDINT</td>
<td>Line index in ipSrcImage1, which is the first line in ipDestImage</td>
</tr>
<tr>
<td>nNumLines</td>
<td>UDINT</td>
<td>The number of lines that should be copied to ipDestImage, starting with nFirstLine in aSrcImageArray[0]. Once the last line of aSrcImageArray[0] was copied, the remaining lines are copied from aSrcImageArray[1] (starting at line index 0) and so on.</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Further information

The function `F_VN_FuseImageArray` is a modified version of the function `F_VN_FuseImages [732]` and can fuse up to 10 images instead of two images.

## Samples

- Fusing several images [136]

## Related functions

- `F_VN_FuseImages [732]` for fusing two images
- `F_VN_FuseImagesArray [734]` for fusing up to 10 images

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.5.21 F_VN_GetImageChannel

Return the specified source image channel as a single-channel image.
Syntax

Definition:

FUNCTION F_VN_GetImageChannel : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    nChannelIndex : UINT;
    hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate image will be created if required.)</td>
</tr>
<tr>
<td>nChannelIndex</td>
<td>UINT</td>
<td>Index of the requested source image channel</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.5.22 F_VN_GetImageHeight

F_VN_GetImageHeight

ipImage ITcVnImage
nHeight Reference To UDINT
hrPrev HRESULT

Return the height of an image. (Alternatively use interface method .GetHeight.)

Syntax

Definition:

FUNCTION F_VN_GetImageHeight : HRESULT
VAR_INPUT
    ipImage : ITcVnImage;
    nHeight : Reference To UDINT;
    hrPrev : HRESULT;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage</td>
<td>ITcVnImage (383)</td>
<td>Image</td>
</tr>
<tr>
<td>nHeight</td>
<td>Reference To UDINT</td>
<td>Return the height of ipImage</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT (135)</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT (135)

## Required License

TC3 Vision Base

## System Requirements

### Development environment
- TwinCAT V3.1.4024.17 or later

### Target platform
- PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU

### PLC libraries to include
- Tc3_Vision 6.4.5.23

### Syntax

#### Definition:

```c
FUNCTION F_VN_GetImageInfo : HRESULT
VAR_INPUT
    ipImage : ITcVnImage;
    stImageInfo : Reference To TcVnImageInfo;
    hrPrev : HRESULT;
END_VAR
```

Gets a struct containing all common meta infos of the image. (Alternatively use interface method .GetImageInfo.)

## 6.4.5.23 F_VN_GetImageInfo

F_VN_GetImageInfo

IPIMAGE _ ITcVnImage
STIMAGEINFO Reference To TcVnImageInfo
HRPREV HRESULT

HRESULT F_VN_GetImageInfo

Return value

HRESULT (135)
**6.4.5.24 F_VN_GetImageWidth**

Return the width of an image. (Alternatively use interface method .GetWidth.)

**Syntax**

Definition:

```c
FUNCTION F_VN_GetImageWidth : HRESULT
VAR_INPUT
    ipImage : ITcVnImage;
    nWidth : Reference To UDINT;
    hrPrev : HRESULT;
END_VAR
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage</td>
<td>ITcVnImage [383]</td>
<td>Image</td>
</tr>
<tr>
<td>nWidth</td>
<td>Reference To UDINT</td>
<td>Return the width of ipImage</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.5.25 F_VN_GetPixel

F_VN_GetPixel


<table>
<thead>
<tr>
<th>ipSrcImage</th>
<th>ITcVnImage</th>
<th>HRESULT</th>
<th>F_VN_GetPixel</th>
</tr>
</thead>
<tbody>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector4_LREAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nX</td>
<td>UDINT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nY</td>
<td>UDINT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

F_VN_GetPixel

Gets a specific pixel.

Syntax

Definition:

FUNCTION F_VN_GetPixel : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    aValue : Reference To TcVnVector4_LREAL;
    nX : UDINT;
    nY : UDINT;
    hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>Returns the pixel value (Unused channels are set to 0.)</td>
</tr>
<tr>
<td>nX</td>
<td>UDINT</td>
<td>x coordinate of the pixel</td>
</tr>
<tr>
<td>nY</td>
<td>UDINT</td>
<td>y coordinate of the pixel</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.5.26 F_VN_GetPixelFormat

F_VN_GetPixelFormat


<table>
<thead>
<tr>
<th>ipImage</th>
<th>ITcVnImage</th>
<th>HRESULT</th>
<th>F_VN_GetPixelFormat</th>
</tr>
</thead>
<tbody>
<tr>
<td>stPixelFormat</td>
<td>Reference To TcVnPixelFormat</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Gets the pixel format of an image. (Alternatively use interface method .GetPixelFormat.)

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_GetPixelFormat : HRESULT
VAR_INPUT
    ipImage : ITcVnImage;
    stPixelFormat : Reference To TcVnPixelFormat;
    hrPrev : HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage</td>
<td>ITcVnImage [383]</td>
<td>Image</td>
</tr>
<tr>
<td>stPixelFormat</td>
<td>Reference To TcVnPixelFormat [221]</td>
<td>Returns a struct describing the pixel format</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.5.27  F_VN_GetRoi

```plaintext
F_VN_GetRoi
ipSrcImage ITcVnImage
stRoi Reference To TcVnRectangle_UDINT
hrPrev HRESULT
HRESULT F_VN_GetRoi
```

Gets the coordinates of the region of interest (ROI) within the image.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_GetRoi : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    stRoi : Reference To TcVnRectangle_UDINT;
    hrPrev : HRESULT;
END_VAR
```
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>stRoi</td>
<td>Reference To TcVnRectangle UDINT [222]</td>
<td>Returns the coordinates of the region of interest</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Further information

The function `F_VN_GetRoi` indicates which Region Of Interest [149] is currently set on the input image `ipSrcImage`. The information is required, for example, if points in the image are to be specified in relation to the total image or if the ROI is to be dynamically enlarged.

Application

The referencing of a point to the coordinates of the original image looks like this, for example:

```plaintext
hr := F_VN_ContourCenterOfMass(ipContour, aCenter, hr);
hr := F_VN_GetRoi(ipImageWork, stRoi, hr);
aCenter[0] := aCenter[0] + stRoi.nX;
```

Related functions

- `F_VN_SetRoi [747]` for setting a Region of Interest
- `F_VN_GetRoi [740]` for retrieving the set Region of Interest
- `F_VN_ResetRoi [742]` for resetting the Region of Interest on the entire image
- `F_VN_CopyImageRegion [719]` for copying an image region

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.5.28  F_VN_MixImageChannels

Mix image channels by copying the specified channels of the source image into the specified channels of the destination image.
## Syntax

### Definition:

```c
FUNCTION F_VN_MixImageChannels : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    ipIndicesFromTo : ITcVnContainer;
    hrPrev : HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [§ 383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [§ 383]</td>
<td>Destination image (An appropriate image will be created if required.)</td>
</tr>
<tr>
<td>ipIndicesFromTo</td>
<td>ITcVnContainer [§ 345]</td>
<td>Index pairs (ContainerType_Vector_TcVnVector2_DINT), specifying which source channel (TcVnVector2_DINT [0]) should be copied to which destination channel (TcVnVector2_DINT [1]).</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [§ 135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

**HRESULT [§ 135]**

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.5.29 **F_VN_ResetRoi**

```c
F_VN_ResetRoi
ipImage  ITcVnImage
hrPrev   HRESULT
```

Reset the region of interest (ROI) of an image. (After this operation, the ROI is set to the entire image.)

### Syntax

#### Definition:

```c
FUNCTION F_VN_ResetRoi : HRESULT
VAR_INPUT
    ipImage : ITcVnImage;
    hrPrev : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage</td>
<td>ITcVnImage[383]</td>
<td>Image</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Further information

The function `F_VN_ResetRoi` resets the Region of Interest [149] of an image. After executing the function, the ROI of the image is at the position [0, 0] and has the size [ImageWidth, ImageHeight].

## Application

The resetting of an ROI looks like this, for example:

```c
hr = F_VN_ResetRoi(ipImageWork, hr);
```

## Related functions

- `F_VN_SetRoi` [747] for setting a Region of Interest
- `F_VN_GetRoi` [740] for retrieving the set Region of Interest
- `F_VN_ResetRoi` [742] for resetting the Region of Interest on the entire image
- `F_VN_CopyImageRegion` [719] for copying an image region

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.5.30 F_VN_SetImageChannel

Sets the specified destination image channel to the values of the specified source image channel.

### Syntax

**Definition:**

```c
FUNCTION F_VN_SetImageChannel : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    nSrcChannelIndex : UINT;
    nDestChannelIndex : UINT;
    hrPrev : HRESULT;
```

```c
F_VN_SetImageChannel(ipSrcImage, ipDestImage, nSrcChannelIndex, nDestChannelIndex, hrPrev);
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate image will be created if required.)</td>
</tr>
<tr>
<td>nSrcChannelIndex</td>
<td>UINT</td>
<td>Index of the source image channel</td>
</tr>
<tr>
<td>nDestChannelIndex</td>
<td>UINT</td>
<td>Index of the destination image channel, which will be replaced by the specified source image channel</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.5.31 **F_VN_SetPixel**

Sets a specific pixel.

**Syntax**

**Definition:**

FUNCTION F_VN_SetPixel : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    aValue : Reference To TcVnVector4_LREAL;
    nX : UDINT;
    nY : UDINT;
    hrPrev : HRESULT;
END_VAR
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipDestImage</td>
<td>ITcVnImage</td>
<td>Destination image</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>The pixel value to set (additional channels are ignored.)</td>
</tr>
<tr>
<td>nX</td>
<td>UDINT</td>
<td>x coordinate of the pixel</td>
</tr>
<tr>
<td>nY</td>
<td>UDINT</td>
<td>y coordinate of the pixel</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.5.32 F_VN_SetPixels**

Sets all pixels of an image to a given value.

**Syntax**

**Definition:**

```
FUNCTION F_VN_SetPixels : HRESULT
VAR_INPUT
    ipDestImage : ITcVnImage;
    aValue     : Reference To TcVnVector4_LREAL;
    hrPrev     : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipDestImage</td>
<td>ITcVnImage</td>
<td>Destination image</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>Pixel value</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.5.33 F_VN_SetPixelsExp

Sets all pixels of an image to a given value. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_SetPixelsExp : HRESULT
VAR_INPUT
    ipDestImage : ITcVnImage;
    aValue      : Reference To TcVnVector4_LREAL;
    ipMask      : ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipDestImage</td>
<td>ITcVnImage [383]</td>
<td>Destination image</td>
</tr>
<tr>
<td>aValue</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Pixel value</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITcVnImage [383]</td>
<td>Mask image</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.5.34 F_VN_SetRoi

**F_VN_SetRoi**

Sets a region of interest (ROI) within an image.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_SetRoi : HRESULT
VAR_INPUT
    nX : UDINT;
    nY : UDINT;
    nWidth : UDINT;
    nHeight : UDINT;
    ipDestImage : ITcVnImage;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nX</td>
<td>UDINT</td>
<td>Left boundary (inclusive 0-based index)</td>
</tr>
<tr>
<td>nY</td>
<td>UDINT</td>
<td>Upper boundary (inclusive 0-based index)</td>
</tr>
<tr>
<td>nWidth</td>
<td>UDINT</td>
<td>ROI width</td>
</tr>
<tr>
<td>nHeight</td>
<td>UDINT</td>
<td>ROI height</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITcVnImage</td>
<td>Destination image</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

**HRESULT [135]**

**Further information**

The function F_VN_SetRoi sets the Region of Interest [149] of an image. The function F_VN_SetRoi_TcVnRectangle_UDINT [748] can be used if the rectangle parameters are not to be transferred individually, but as a structure of the type TcVnRectangle_UDINT [222].

**Application**

The setting of an ROI with the size [240, 120] at the position [50, 50] looks like this, for example:

```c
hr := F_VN_SetRoi(50, 50, 240, 120, ipImageWork, hr);
```
Related functions

- F_VN_SetRoi [747] for setting a Region of Interest
- F_VN_GetRoi [740] for retrieving the set Region of Interest
- F_VN_ResetRoi [742] for resetting the Region of Interest on the entire image
- F_VN_CopyImageRegion [719] for copying an image region

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.5.35  F_VN_SetRoi_TcVnRectangle_UDINT

Sets a region of interest (ROI) within an image.

Syntax

**Definition:**

```plaintext
FUNCTION F_VN_SetRoi_TcVnRectangle_UDINT : HRESULT
VAR_INPUT
    stRoi : Reference To TcVnRectangle_UDINT;
    ipDestImage : ITcVnImage;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stRoi</td>
<td>Reference To TcVnRectangle_UDINT [222]</td>
<td>Region of interest</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITcVnImage [383]</td>
<td>Destination image</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

Further information

Like the function F_VN_SetRoi [747], the function F_VN_SetRoi_TcVnRectangle_UDINT sets a Region of Interest [149]. However, it doesn’t accept the rectangle parameters individually, but as a structure of the type TcVnRectangle_UDINT [222].
This makes it possible, for example, to set the ROI directly with the result of the function `F_VN_UprightBoundingRectangle` [777].

### Application

The setting of an ROI around a found contour [146] looks like this, for example:

```plaintext
hr := F_VN_UprightBoundingRectangle(ipContour, stRoi, hr);
hr := F_VN_SetRoi_TcVnRectangle_UDINT(stRoi, iplImageWork, hr);
```

### Related functions

- [F_VN_SetRoi](#) for setting a Region of Interest
- [F_VN_GetRoi](#) for retrieving the set Region of Interest
- [F_VN_ResetRoi](#) for resetting the Region of Interest on the entire image
- [F_VN_CopyImageRegion](#) for copying an image region

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.5.36  F_VN_SplitImageChannels

#### Definition:

```plaintext
FUNCTION F_VN_SplitImageChannels : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    pDestImages : PVOID;
    nArraySize : UINT;
    hrPrev : HRESULT;
END_VAR
VAR_OUTPUT
    nImageChannels : UINT;
END_VAR
```

Split a multi-channel image into multiple single channel images (1 for each source image channel).

### Syntax

- **ipSrcImage**: `ITcVnImage` — The multi-channel image to split.
- **pDestImages**: `PVOID` — Pointer to an array of single channel images.
- **nArraySize**: `UINT` — Number of array elements.
- **hrPrev**: `HRESULT` — Error code from previous function.
- **nImageChannels**: `UINT` — Number of single-channel images created.

```plaintext
F_VN_SplitImageChannels
ipSrcImage    ITcVnImage
pDestImages   PVOID
nArraySize    UINT
hrPrev        HRESULT
```

Split a multi-channel image into multiple single channel images (1 for each source image channel).
API reference

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvImage [383]</td>
<td>Source image with multiple channels</td>
</tr>
<tr>
<td>pDestImages</td>
<td>PVOID</td>
<td>Pointer to an array of ITCvImage (appropriate destination images will be created if required)</td>
</tr>
<tr>
<td>nArraySize</td>
<td>UINT</td>
<td>Number of pDestImages array elements (array must be &gt;= ipSourceImage channels)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nImageChannels</td>
<td>UINT</td>
<td>Actual source image channels</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.5.37 **F_VN_TransformIntoDisplayableImage**

Transform an image into a displayable image. The source image will be released and zeroed while existing destination images will be released and overwritten with the function result. The source image must not be used anywhere else and the transformation is very fast. If you want to use ipSrcImage after this function call, use F_VN_CopyIntoDisplayableImage instead.

**Syntax**

**Definition:**

```
FUNCTION F_VN_TransformIntoDisplayableImage : HRESULT
VAR_INPUT
    ipSrcImage : Reference To ITCvImage;
    ipDestImage : Reference To ITCvDisplayableImage;
    hrPrev : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>Reference To ITcVnImage</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnDisplayableImage</td>
<td>Returns the displayable image</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Further information

For further details, see Displayable images [142].

### NOTE

**Use with Ads Communicator object**

Note the case where the image to be transformed originates from a camera object whose Ads Communicator object is active! It may then be the case that the image is not internally released in the object and therefore cannot be transformed into a displayable image. In this case, use F_VN_CopyIntoDisplayableImage [721] or F_VN_TransformIntoDisplayableImageExp [751] with bAllowDeepCopy := TRUE.

### Related functions

- F_VN_CopyIntoDisplayableImage [721]
- F_VN_TransformIntoDisplayableImageExp [751]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.5.38 **F_VN_TransformIntoDisplayableImageExp**

Transform an image into a displayable image. The source image will be released and zeroed while existing destination images will be released and overwritten with the function result. If bAllowDeepCopy equals false, the source image must not be used anywhere else and the transformation is always very fast. Otherwise, if bAllowDeepCopy equals true, a deep copy of the source image might be created if required, which will result in a longer execution time. If you want to use ipSrcImage after this function call, use F_VN_CopyIntoDisplayableImage instead. (expert function)
Syntax

Definition:

FUNCTION F_VN_TransformIntoDisplayableImageExp : HRESULT
VAR_INPUT
  ipSrcImage : Reference To ITcVnImage;
  ipDestImage : Reference To ITcVnDisplayableImage;
  bAllowDeepCopy : BOOL;
  hrPrev : HRESULT;
END_VAR


Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>Reference To ITcVnImage</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnDisplayableImage</td>
<td>Returns the displayable image</td>
</tr>
<tr>
<td>bAllowDeepCopy</td>
<td>BOOL</td>
<td>Allow deep image copies, if required</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Further information

For further details, see Displayable images [142].

Parameter

Permission for deep copy

The permission (TRUE) for a deep copy can be given with the parameter bAllowDeepCopy. This is relevant if the image to be transformed originates from a camera object whose Ads Communicator object [118] is activated and the image therefore can be internally referenced.

The standard version F_VN_TransformIntoDisplayableImage [750] can also be used if no permission is to be given (FALSE).

Related functions

- F_VN_CopyIntoDisplayableImage [721]
- F_VN_TransformIntoDisplayableImage [750]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

TC7000 - TF7300
6.4.6  Code Reading

This group contains functions for detecting and reading 1D and 2D codes.

Functions

1D codes

- F_VN_ReadBarcode(Exp) [756]
  - CodaBar
  - Code39
  - Code93
  - Code128
  - EAN8
  - EAN13
  - ITF
  - UPCA
  - UPCE
  - Code39Extended

- F_VN_ReadPharmaCode(Exp) [764]

2D codes

- F_VN_ReadDataMatrixCode(Exp) [761]
- F_VN_ReadQRCode(Exp) [768]

Code requirements

In general, a module size of at least three pixels is required; more than three pixels are recommended for more stable results. The image should be sharp and have a high contrast. In addition, a code orientation of approx. 0° or 90° is recommended so that the code is parallel to an image axis.

Interpretation of the HRESULT

For the Code Reading functions the HRESULT [135] return values are used as follows:

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>16#000</td>
<td>S_OK</td>
<td>Function was successfully executed and all expected codes were found (at least one code for the standard functions and nCodeNumber for the expert functions).</td>
</tr>
<tr>
<td>16#001</td>
<td>S_FALSE</td>
<td>Function was successfully executed and all expected codes were found (at least one code for the standard functions and nCodeNumber for the expert functions).</td>
</tr>
<tr>
<td>16#256</td>
<td>S_WATCHDOGTIMEOUT</td>
<td>Function was aborted by the watchdog. Some Code Reading functions can return partial results (see individual function descriptions).</td>
</tr>
<tr>
<td>16#7xx</td>
<td>All error codes</td>
<td>Function was not executed successfully. See also: ADS Return Codes [1486]</td>
</tr>
</tbody>
</table>
Therefore, the usual check with `SUCCEEDED()` is not sufficient to deduce a found code and thus existing data in `ipDecodedData`. The following query can be used for this purpose:

```c
IF hr = S_OK THEN
    // Export Code into String
    hr := F_VN_ExportSubContainer_String(ipDecodedCode, 0, sCodeAsString, 255, hr);
    // Use sCodeAsString
ELSIF SUCCEEDED(hr) THEN
    // Process partial results
ELSE
    // Error handling
END_IF
```

### Search strategies for 2D codes

For 2D codes the search algorithms are applied to different variants of the input image in order to recognize as many codes as possible. Both inversion and mirroring of the image are used to generate various versions. The decision as to which variant is tested is called the search strategy. So that you have greater control over the runtime of the functions, this search strategy can be configured. The enum `ETcVn2dCodeSearchStrategy` is available for this purpose. With this you can specify for each possible image transformation (inversion and mirroring) whether it is to be applied and whether the original image or the transformed image should be checked first for codes.

The function parameter `eSearchStrategy` defines precisely this search strategy for corresponding code reading expert functions. The tables below show the different combinations:

**Table 1: Inversion**

<table>
<thead>
<tr>
<th>eSearchStrategy</th>
<th>Original image</th>
<th>Inverted image</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_CSS_ONLY_NOT_INVERTED</td>
<td>1.</td>
<td>-</td>
</tr>
<tr>
<td>TCVN_CSS_FIRST_NOT_INVERTED</td>
<td>1.</td>
<td>2.</td>
</tr>
<tr>
<td>TCVN_CSS_ONLY_INVERTED</td>
<td>-</td>
<td>1.</td>
</tr>
<tr>
<td>TCVN_CSS_FIRST_INVERTED</td>
<td>2.</td>
<td>1.</td>
</tr>
</tbody>
</table>

**Table 2: Mirroring**

<table>
<thead>
<tr>
<th>eSearchStrategy</th>
<th>Original image</th>
<th>Mirrored image</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCVN_CSS_ONLY_NOT_FLIPPED</td>
<td>1.</td>
<td>-</td>
</tr>
<tr>
<td>TCVN_CSS_FIRST_NOT_FLIPPED</td>
<td>1.</td>
<td>2.</td>
</tr>
<tr>
<td>TCVN_CSS_ONLY_FLIPPED</td>
<td>-</td>
<td>1.</td>
</tr>
<tr>
<td>TCVN_CSS_FIRST_FLIPPED</td>
<td>2.</td>
<td>1.</td>
</tr>
</tbody>
</table>

A search strategy can be defined for each type of transformation. The search strategies of the different search strategies can be linked as follows:

```c
eSearchStrategy := TCVN_CSS_ONLY_FLIPPED + TCVN_CSS_FIRST_INVERTED;
```

The `TCVN_CSS_DEFAULT` option can be used to select a default setting that varies depending on the code type.

**Linking of search strategies not possible**

The linking of several search strategies for the same transformation type is not permissible. Only one strategy for inversion and one for mirroring can be linked to each other.
6.4.6.1 F_VN_ReadBarcode

Detect and interpret a 1d barcode within the provided image. Can be canceled by Watchdog.

Syntax

Definition:
FUNCTION F_VN_ReadBarcode : HRESULT
VAR_INPUT
  ipSrcImage : ITCvnImage;
  ipDecodedData : Reference To ITcVnContainer;
  eBarcodeType : UDINT;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvnImage [383]</td>
<td>Source image (USINT elements, 1 channel or 3 channel (3 channel input is expected to be RGB and internally converted to Gray))</td>
</tr>
<tr>
<td>ipDecodedData</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns the decoded code (ContainerType_Vector_String_SINT)</td>
</tr>
<tr>
<td>eBarcodeType</td>
<td>UDINT</td>
<td>Types of barcode to search for (ETcVnBarcodeType)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Further information

The function F_VN_ReadBarcode detects a barcode in the image and reads it.

Algorithm

The function first searches for horizontally aligned codes and then for vertically aligned codes in the image.

Parameter

Input image

The input image ipSrcImage must be a 1-channel or 3-channel image with the element type USINT (8-bit). 3-channel images will be interpreted as RGB and converted internally into grayscale images.

Read data

The data read from the code are returned as a string in the container ipDecodedData. The container is of the type ContainerType_Vector_String_SINT.
Barcode type

The parameter `eBarcodeType` defines the barcode types to be searched for in the image. All barcode types in the enum `ETcVnBarcodeType` are supported. It is possible to select more than one type. It is recommended to restrict the selection to the types actually occurring in the end application in order to improve the runtime.

```c
eBarcodeType := TCVN_BT_EAN13 + TCVN_BT_CODE128;
```

Use `TCVN_BT_ANY` to search for all available barcode types.

Expert parameters

The expert version `F_VN_ReadBarcodeExp` contains additional parameters.

Read data from container

If the code was successfully found and decoded, the code content is returned in the container `ipDecodedData` and can be exported to a string using the function `F_VN_ExportSubContainer_String`.

```c
hr := F_VN_ExportSubContainer_String( ipDecodedData, 0, sText, nMaxLength, hr);
```

Samples

- EAN-13 Barcode Reading
- Result evaluation during Code Reading

Related functions

- `F_VN_ReadBarcode`
- `F_VN_ReadPharmaCode`
- `F_VN_ReadDataMatrixCode`
- `F_VN_ReadQRCode`

Required License

TC3 Vision Code Reading

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.6.2 F_VN_ReadBarcodeExp

```c
F_VN_ReadBarcodeExp
```

Detect and interpret a 1d barcode within the provided image. Can be canceled by Watchdog.
**Syntax**

**Definition:**

```c
FUNCTION F_VN_ReadBarcodeExp : HRESULT
VAR_INPUT
    ipSrcImage : ITCvNImage;
    ipDecodedData : Reference To ITCvNContainer;
    ipContours : Reference To ITCvNContainer;
    eBarcodeType : UDINT;
    nCodeNumber : DINT;
    eSearchDirection : ETcVnBarcodeSearchDirection;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvNImage [383]</td>
<td>Source image (USINT elements, 1 channel or 3 channel (3 channel input is expected to be RGB and internally converted to Gray))</td>
</tr>
<tr>
<td>ipDecodedData</td>
<td>Reference To ITCvNContainer [345]</td>
<td>Returns the decoded code (ContainerType_Vector_String_SINT)</td>
</tr>
<tr>
<td>ipContours</td>
<td>Reference To ITCvNContainer [345]</td>
<td>Returns the code positions as contours (ContainerType_Vector_Vector_TcVnPoint2_DINT)</td>
</tr>
<tr>
<td>eBarcodeType</td>
<td>UDINT</td>
<td>Types of barcode to search for (ETcVnBarcodeType)</td>
</tr>
<tr>
<td>nCodeNumber</td>
<td>DINT</td>
<td>Number of 1d barcode that should be detected within the provided image (currently only 1 supported).</td>
</tr>
<tr>
<td>eSearchDirection</td>
<td>ETcVnBarcodeSearchDirection [155]</td>
<td>Barcode search direction.</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Further information**

The function **F_VN_ReadBarcodeExp** is the expert version of **F_VN_ReadBarcode [755]**. It contains additional parameters for returning the code position and for setting the search direction.

**Parameter**

**Input image**

The input image `ipSrcImage` must be a 1-channel or 3-channel image with the element type USINT (8-bit). 3-channel images will be interpreted as RGB and converted internally into grayscale images.

**Read data**

The data read from the code are returned as a string in the container `ipDecodedData`. The container is of the type `ContainerType_Vector_String_SINT`.
Barcode type

The parameter `eBarcodeType` defines the barcode types to be searched for in the image. All barcode types in the enum `ETcVnBarcodeType` are supported. It is possible to select more than one type. It is recommended to restrict the selection to the types actually occurring in the end application in order to improve the runtime.

```plaintext
eBarcodeType := TCVN_BT_EAN13 + TCVN_BT_CODE128;
```

Use `TCVN_BT_ANY` to search for all available barcode types.

Position of the code

The parameter `ipContour` returns the code position that is found as a contour.

Number of codes

The parameter `nCodeNumber` defines how many codes to search for.

- **Search for more than one barcode not supported**
  - Searching for several barcodes in one image is currently not supported. The parameter `nCodeNumber` therefore has to be set to 1.

Search direction

The search direction can be specified via `eSearchDirection`:

- `TCVN_BSD_ANY` first searches in horizontal direction, then in vertical direction.
- `TCVN_BSD_HORIZONTAL` only searches in horizontal direction.
- `TCVN_BSD_VERTICAL` only searches in vertical direction.

Samples

- [EAN-13 Barcode Reading](#)
- [Result evaluation during Code Reading](#)

Related functions

- [F_VN_ReadBarcode](#)
- [F_VN_ReadPharmaCode](#)
- [F_VN_ReadDataMatrixCode](#)
- [F_VN_ReadQRCode](#)

Required License

TC3 Vision Code Reading

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.6.3  F_VN_ReadDataMatrixCode

F_VN_ReadDataMatrixCode

<table>
<thead>
<tr>
<th>Input</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image (USINT elements, 1 or 3 channels)</td>
</tr>
<tr>
<td>ipDecodedData</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns the decoded code (ContainerType_Vector_String_SINT)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Detect and interpret a data matrix code within the provided image. Can return partial results when canceled by Watchdog.

Syntax

Definition:

FUNCTION F_VN_ReadDataMatrixCode : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDecodedData : Reference To ITcVnContainer;
    hrPrev : HRESULT;
END_VAR

Further information

The data matrix barcode according to IEC 16022 is a 2D code that facilitates storage of large data quantities with minimum space requirements. It is therefore often used for component and product marking.

The function F_VN_ReadDataMatrixCode can recognize and decode this code type in the image. This is based on one code per image. If several Data Matrix Codes are to be recognized and decoded in one image, the function F_VN_ReadDataMatrixCodeExp [761] must be used.

Data Matrix Code structure

The Data Matrix Code consists of rectangles or dots that make up a square or rectangular symbol, which in turn can be divided into three areas:

1. Finder Pattern
   The solid-line locator (L) is used to locate and align the code, while the pattern-line locator (P) is used to determine the cell size and arrangement.
2. Data area
   The data area is surrounded by the Finder Pattern.

3. Quiet zone
   The quiet zone is an area of at least 1.5 cells around the Finder Pattern in the background color of the code.

Flaws / damage in the Finder Pattern or in the quiet zone can prevent detection of the code.

Error correction

Data matrix code ECC 200 (ECC = Error Correction Code) is the generally accepted version due to its better error correction. In addition to the actual data, they are coded redundantly based on the Reed-Solomon code. Data matrix codes of this version can be decoded by the function. In case of damage an error correction can be performed.

Code search

The F_VN_ReadDataMatrixCode function searches for a Data Matrix Code in the image, starting from the center line of the image.

Read data from container

If the code was successfully found and decoded, the code content is returned in the container ipDecodedData and can be exported to a string using the function F_VN_ExportSubContainer_String.

   hr := F_VN_ExportSubContainer_String( ipDecodedData, 0, sText, nMaxLength, hr);

HRESULT

Like all TwinCAT Vision API functions, this function also returns an HRESULT to indicate whether the execution was successful or not. In the event of success, a distinction can be made between the following success codes in the subsequent handling of the results. To distinguish between these cases, the HRESULT variable can be compared directly with \texttt{S\_OK} or \texttt{S\_FALSE}.

\begin{table}[ht]
\centering
\begin{tabular}{|c|c|l|}
\hline
Code & Name & Description \\
\hline
16#000 & S\_OK & The function was executed successfully. The specified number of codes was found in the image. \\
16#001 & S\_FALSE & The function was executed successfully. Fewer codes than indicated in the image were found. \\
16#256 & S\_WATCHDOG TIMEOUT & The function was aborted by the watchdog. \\
\hline
\end{tabular}
\end{table}

Samples

- Data Matrix Code Reading
- Result evaluation during Code Reading

Related functions

- F_VN_ReadDataMatrixCodeExp
- F_VN_ReadBarcode or F_VN_ReadBarcodeExp
- F_VN_ReadPharmaCode or F_VN_ReadPharmaCodeExp
- F_VN_ReadQRCode or F_VN_ReadQRCodeExp

Required License

TC3 Vision Code Reading
## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.6.4  F_VN_ReadDataMatrixCodeExp

Detect and interpret a data matrix code within the provided image. Can return partial results when canceled by Watchdog.

**Syntax**

**Definition:**
```
FUNCTION F_VN_ReadDataMatrixCodeExp : HRESULT
VAR_INPUT
   ipSrcImage : ITCvImage;
   ipDecodedData : Reference To ITCvContainer;
   ipContours : Reference To ITCvContainer;
   nCodeNumber : DINT;
   eSearchStrategy : UDINT;
   hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvImage [383]</td>
<td>Source image (USINT elements, 1 or 3 channels)</td>
</tr>
<tr>
<td>ipDecodedData</td>
<td>Reference To ITCvContainer [345]</td>
<td>Returns the decoded code (ContainerType_Vector_String_SINT)</td>
</tr>
<tr>
<td>ipContours</td>
<td>Reference To ITCvContainer [345]</td>
<td>Returns the code positions as contours (ContainerType_Vector_Vector_TcVnPoint2_DINT)</td>
</tr>
<tr>
<td>nCodeNumber</td>
<td>DINT</td>
<td>Number of data matrix codes that should be detected within the provided image. (If set to -1, it is tried to detect all data matrix codes.)</td>
</tr>
<tr>
<td>eSearchStrategy</td>
<td>UDINT</td>
<td>Used search strategy (ETcVn2dCodeSearchStrategy)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Further information**

The function F_VN_ReadDataMatrixCodeExp corresponds to the function F_VN_ReadDataMatrixCode [759] extended by the following parameters.

ipContours
The parameter `ipContour` returns the code positions that are found as a contour.

**nCodeNumber**

The parameter `nCodeNumber` defines how many codes to search for. At `-1` the search continues until no further code is found in the image or a watchdog aborts the function.

**eSearchStrategy**

The parameter `eSearchStrategy` defines the search strategies for the code search. The standard option `TCVN_CSS_DEFAULT` leads to the search strategy `TCVN_CSS_FIRST_NOT_FLIPPED`. For a description of the search strategies see Code Reading search strategies [745].

### Search strategies

Currently, this function only supports the settings of the search strategy for mirroring. Inversion is always performed automatically if not all expected codes are found in the original image.

### Samples

- Data Matrix Code Reading [1372]
- Result evaluation during Code Reading [1377]

### Related functions

- F_VN_ReadDataMatrixCode [759]
- F_VN_ReadBarcode [755] or F_VN_ReadBarcodeExp [756]
- F_VN_ReadPharmaCode [762] or F_VN_ReadPharmaCodeExp [764]
- F_VN_ReadQRCode [766] or F_VN_ReadQRCodeExp [768]

### Required License

TC3 Vision Code Reading

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.6.5 F_VN_ReadPharmaCode

Detect and interpret a pharma code within the provided image. Can be canceled by Watchdog. Can use available TwinCAT Job Tasks for executing parallel code regions.

#### Syntax

**Definition:**

```c
FUNCTION F_VN_ReadPharmaCode : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDecodedData : Reference To ITcVnContainer;
    hrPrev : HRESULT;
END_VAR
```

F_VN_ReadPharmaCode

ipSrcImage  ITcVnImage HRESULT F_VN_ReadPharmaCode
ipDecodedData  Reference To ITcVnContainer
hrPrev  HRESULT
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image (1 channel (all element types) or 3 channel with elements of type TCVN_ET_USINT, TCVN_ET_UINT or TCVN_ET_REAL (3 channel input is expected to be RGB and internally converted to Gray))</td>
</tr>
<tr>
<td>ipDecodedData</td>
<td>Reference To ITcVnContainer</td>
<td>Returns the decoded code (ContainerType_Vector_String_SINT)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [383]

**Further information**

The function F_VN_ReadPharmaCode detects and reads a pharmacode in the input image ipSrcImage. The Pharmacode is a barcode standard. It is designed for good legibility but can only encode integers in the range 3 to 131070. The aim of this standard is high reading speed with few misreadings.

---

**Do not mix up pharmacodes**

The Pharmacode Standard should not be confused with the Italian or Swiss pharmacodes. These only designate drug identification numbers with a Code39 representation.

---

**Structure of Pharmacodes**

A Pharmacode consists of thick and thin bars (dark) and gaps between them (light). For good legibility, sufficient size and contrast of the code in the image are recommended. Decoding is based on the following table:

<table>
<thead>
<tr>
<th>Position</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thin line</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Thick Line</td>
<td>256</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

Detailed information on the specification of Pharmacodes can be found at:

- RC Electronica specification
- Laetus Pharmacode guide

**Parameter**

**Input image**

The input image ipSrcImage must have 1 or 3 channels. 3-channel images must be of the type USINT, UINT or REAL and must be in the RGB color space. They are converted internally into a grayscale image.

**Read data (Return value)**

The data read from the code are returned as a string in the container ipDecodedData. The container is of the type ContainerType_Vector_String_SINT.

**Expert parameters**

Read data from container

If the code was successfully found and decoded, the code content is returned in the container `ipDecodedData` and can be exported to a string using the function `F_VN_ExportSubContainer_String`.

```c
hr := F_VN_ExportSubContainer_String(ipDecodedData, 0, sText, nMaxLength, hr);
```

### Samples

- Pharma-Code Reading [1373]
- Result evaluation during Code Reading [1377]

### Related functions

- `F_VN_ReadPharmaCode` [762] for reading pharmacodes
- `F_VN_ReadBarcode` [755] for reading barcodes
- `F_VN_ReadDataMatrixCode` [759] for reading data matrix codes
- `F_VN_ReadQRCode` [766] for reading QR codes

### Required License

TC3 Vision Code Reading

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.6.6 F_VN_ReadPharmaCodeExp

Detect and interpret a pharma code within the provided image. Can be canceled by Watchdog. Can use available TwinCAT Job Tasks for executing parallel code regions.

#### Syntax

**Definition:**

```c
FUNCTION F_VN_ReadPharmaCodeExp : HRESULT
VAR_INPUT
    ipSrcImage ITCvImage
    ipDecodedData Reference To ITCvntContainer
    ipContours Reference To ITCvntContainer
    nCodeNumber DINT
    nMinBarNumber INT
    hrPrev HRESULT
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image (1 channel (all element types) or 3 channel with elements of type TCVN_ET_USINT, TCVN_ET_UINT or TCVN_ET_REAL (3 channel input is expected to be RGB and internally converted to Gray))</td>
</tr>
<tr>
<td>ipDecodedData</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns the decoded code (ContainerType_Vector_String_SINT)</td>
</tr>
<tr>
<td>ipContours</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns the code positions as contours (ContainerType_Vector_Vector_TcVnPoint2_DINT)</td>
</tr>
<tr>
<td>nCodeNumber</td>
<td>DINT</td>
<td>Number of pharma codes that should be detected within the provided image (currently only 1 supported).</td>
</tr>
<tr>
<td>nMinBarNumber</td>
<td>INT</td>
<td>Minimal number of (dark) bars that codes must have.</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Further information

This Expert function extends the F_VN_ReadPharmaCode [762] function as follows:

- Return of the position of found codes via the parameter `ipContours`
- Specifies the number of codes to be found in the image. The search is terminated once `nCodeNumber` or more codes have been found in the image. If less than `nCodeNumber` codes are found, the function returns S_FALSE.
- Minimum number of dark bars that the code must have to be found by the function via the parameter `nMinBarNumber`.

## Samples

- Pharma-Code Reading [1373]
- Result evaluation during Code Reading [1377]

## Related functions

- F_VN_ReadPharmaCode [762]
- F_VN_ReadBarcode [755] or F_VN_ReadBarcodeExp [757]
- F_VN_ReadDataMatrixCode [759] or F_VN_ReadDataMatrixCodeExp [761]
- F_VN_ReadQRCode [766] or F_VN_ReadQRCodeExp [768]

## Required License

TC3 Vision Code Reading

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.6.7  F_VN_ReadQRCode

Detect and interpret a QR code within the provided image. Can be canceled by Watchdog.

**Syntax**

Definition:

```plaintext
FUNCTION F_VN_ReadQRCode : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDecodedData : Reference To ITcVnContainer;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image (USINT elements, 1 channel or 3 channel (3 channel input is expected to be RGB and internally converted to Gray))</td>
</tr>
<tr>
<td>ipDecodedData</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns the decoded code (ContainerType_Vector_String_SINT)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Further information**

The QR code (Quick-Response code) according to IEC 18004 is a 2D code, which is particularly common in Germany in the field of mobile tagging. Thanks to the option to encode Kanji/Kana characters, the QR code has also become established in the industry in Asia.

The F_VN_ReadQRCode function can recognize and decode this code type in the image. This is based on one code per image.
QR code structure

1. Position markings
2. Alignment mark
3. Timing rows
4. Version information
5. Data format
6. Data field
7. Quiet zone, at least 4 cells wide

Flaws / damage can prevent detection and decoding.

Code search

The function F_VN_ReadQRCode searches line by line for a QR code in the image from top left to bottom right.

Read data from container

If the code was successfully found and decoded, the code content is returned in the container ipDecodedData and can be exported to a string using the function F_VN_ExportSubContainer_String.

```csharp
hr := F_VN_ExportSubContainer_String(ipDecodedData, 0, sText, nMaxLength, hr);
```

HRESULT

Like all TwinCAT Vision API functions, this function also returns an HRESULT to indicate whether the execution was successful or not. In the event of success, a distinction can be made between the following success codes in the subsequent handling of the results. To distinguish between these cases, the HRESULT variable can be compared directly with S_OK or S_FALSE.
### Code Reference

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>16#000</td>
<td>S_OK</td>
<td>The function was executed successfully. The specified number of codes was found in the image.</td>
</tr>
<tr>
<td>16#001</td>
<td>S_FALSE</td>
<td>The function was executed successfully. Fewer codes than indicated in the image were found.</td>
</tr>
<tr>
<td>16#256</td>
<td>S_WATCHDOG</td>
<td>The function was aborted by the watchdog.</td>
</tr>
</tbody>
</table>

### Samples

- QR-Code Reading [1375]
- Result evaluation during Code Reading [1377]

### Related functions

- F_VN_ReadQRCodeExp [768]
- F_VN_ReadBarcode [755] or F_VN_ReadBarcodeExp [756]
- F_VN_ReadDataMatrixCode [759] or F_VN_ReadDataMatrixCodeExp [761]
- F_VN_ReadPharmaCode [762] or F_VN_ReadPharmaCodeExp [764]

### Required License

TC3 Vision Code Reading

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.6.8 F_VN_ReadQRCodeExp

Detect and interpret a QR code within the provided image. Can be canceled by Watchdog.

#### Syntax

Definition:

```c
FUNCTION F_VN_ReadQRCodeExp : HRESULT
VAR_INPUT
  ipSrcImage : ITCvImage;
  ipDecodedData : Reference To ITCvContainer;
  ipContours : Reference To ITCvContainer;
  nCodeNumber : DINT;
  eSearchStrategy : UDINT;
  hrPrev : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image (USINT elements, 1 channel or 3 channel (3 channel input is expected to be RGB and internally converted to Gray))</td>
</tr>
<tr>
<td>ipDecodedData</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns the decoded code (ContainerType_Vector_String_SINT)</td>
</tr>
<tr>
<td>ipContours</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns the code positions as contours (ContainerType_Vector_Vector_TcVnPoint2_DINT)</td>
</tr>
<tr>
<td>nCodeNumber</td>
<td>DINT</td>
<td>Number of QR codes that should be detected within the provided image (currently only 1 supported).</td>
</tr>
<tr>
<td>eSearchStrategy</td>
<td>UDINT</td>
<td>Used search strategy (ETcVn2dCodeSearchStrategy)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Further information

### NOTE

**Floating point exceptions**

This function can unnecessarily cause errors if the option Floating point exceptions of the executing PLC task [58] is active. Therefore, deactivate this option.

The function F_VN_ReadQRCodeExp corresponds to the function F_VN_ReadQRCode [766], extended by the following parameters.

**ipContour**

The parameter `ipContour` returns the code position that is found as a contour.

**nCodeNumber**

The parameter `nCodeNumber` defines how many codes to search for.

* Searching for more than one QR code not currently supported

Searching for several QR codes in one image is currently not supported. The parameter `nCodeNumber` therefore has to be set to 1.

**eSearchStrategy**

The parameter `eSearchStrategy` defines the search strategies for the code search. The standard option `TCVN_CSS_DEFAULT` leads to the search strategies `TCVN_CSS_FIRST_NOT_INVERTED` and `TCVN_CSS_FIRST_NOT_FLIPPED`. For a description of the search strategies see Code Reading search strategies [754].

### Samples

- QR-Code Reading [1375]
- Result evaluation during Code Reading [1377]

### Related functions

- F_VN_ReadQRCode [766]
API reference

- `F_VN_ReadBarcode` [755] or `F_VN_ReadBarcodeExp` [756]
- `F_VN_ReadDataMatrixCode` [759] or `F_VN_ReadDataMatrixCodeExp` [761]
- `F_VN_ReadPharmaCode` [762] or `F_VN_ReadPharmaCodeExp` [764]

Required License
TC3 Vision Code Reading

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7 Container Statistics

This group contains functions for statistical analysis of containers [143].

Functions

With Euclidean norm

- `F_VN_ContainerAverage` [840]
- `F_VN_ContainerAverageVariance` [845]
- `F_VN_MaxElement` [849]
- `F_VN_MedianElement` [850]
- `F_VN_MinElement` [851]

Element-wise

- `F_VN_ContainerAverageElementwise` [841]
- `F_VN_ContainerAverageVarianceElementwise` [846]
- `F_VN_MaxElementElementwise` [770]
- `F_VN_MedianElementElementwise` [794]
- `F_VN_MinElementElementwise` [817]

6.4.7.1 `F_VN_MaxElementElementwise`

6.4.7.1.1 `F_VN_MaxElementElementwise_DINT`

```csharp
FUNCTION F_VN_MaxElementElementwise_DINT : HRESULT
VAR_INPUT
  ipContainer : ITCvNContainer;
  nMax : Reference To DINT;
  hrPrev : HRESULT;
```

Gets the element wise maximum container element.

Syntax

Definition:

```csharp
FUNCTION F_VN_MaxElementElementwise_DINT : HRESULT
VAR_INPUT
  ipContainer : ITCvNContainer;
  nMax : Reference To DINT;
  hrPrev : HRESULT;
```
Relevant License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.7.1.2 F_VN_MaxElementElementwise_INT

![FUNCTION_BLOCK_F_VN_MaxElementElementwise_INT](image)

<table>
<thead>
<tr>
<th>ipContainer</th>
<th>ITcVnContainer</th>
<th>hrPrev</th>
<th>HRESULT</th>
</tr>
</thead>
</table>

Gets the element wise maximum container element.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_MaxElementElementwise_INT : HRESULT
VAR_INPUT
  ipContainer : ITcVnContainer;
  nMax : Reference To INT;
  hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer <img src="image" alt="345" /></td>
<td>Source container</td>
</tr>
<tr>
<td>nMax</td>
<td>Reference To INT</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT <img src="image" alt="135" /></td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.1.3  **F_VN_MAXELEMENTELEMENTWISE_LREAL**

Gets the element wise maximum container element.

Syntax

Definition:

```cpp
FUNCTION F_VN_MAXELEMENTELEMENTWISE_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    fMax : Reference To LREAL;
    hrPrev : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>fMax</td>
<td>Reference To LREAL</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.7.1.4  F_VN_MaxElementElementwise_REAL

Gets the element wise maximum container element.

Syntax

Definition:

FUNCTION F_VN_MaxElementElementwise_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    fMax        : Reference To REAL;
    hrPrev      : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Source container</td>
</tr>
<tr>
<td>fMax</td>
<td>Reference To REAL</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.1.5  F_VN_MaxElementElementwise_SINT

Gets the element wise maximum container element.

Syntax

Definition:

FUNCTION F_VN_MaxElementElementwise_SINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>nMax</td>
<td>Reference To SINT</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.7.1.6 F_VN_MaxElementElementwise_TcVnPoint2_DINT

```
FUNCTION F_VN_MaxElementElementwise_TcVnPoint2_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aMax        : Reference To TcVnPoint2_DINT;
    hrPrev      : HRESULT;
END_VAR
```

Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_TcVnPoint2_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aMax        : Reference To TcVnPoint2_DINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMax</td>
<td>Reference To TcVnPoint2_DINT</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.7.1.7 F_VN_MaxElementElementwise_TcVnPoint2_LREAL

<table>
<thead>
<tr>
<th>IPContainer: ITcVnContainer</th>
<th>HRESULT: F_VN_MaxElementElementwise_TcVnPoint2_LREAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>aMax: Reference To TcVnPoint2_LREAL</td>
<td>hrPrev: HRESULT</td>
</tr>
</tbody>
</table>

Gets the element wise maximum container element.

**Syntax**

**Definition:**

FUNCTION F_VN_MaxElementElementwise_TcVnPoint2_LREAL : HRESULT

VAR_INPUT

- ipContainer: ITcVnContainer;
- aMax: Reference To TcVnPoint2_LREAL;
- hrPrev: HRESULT;

END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMax</td>
<td>Reference To TcVnPoint2_LREAL [151]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.7.1.8 **F_VN_MaxElementElementwise_TcVnPoint2_REAL**

**Syntax**

**Definition:**

FUNCTION F_VN_MaxElementElementwise_TcVnPoint2_REAL : HRESULT  
VAR_INPUT  
  ipContainer : ITCvNContainer;  
  aMax        : Reference To TcVnPoint2_REAL;  
  hrPrev      : HRESULT;  
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMax</td>
<td>Reference To TcVnPoint2_REAL [151]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.1.9 **F_VN_MaxElementElementwise_TcVnPoint3_LREAL**

**Syntax**

**Definition:**

FUNCTION F_VN_MaxElementElementwise_TcVnPoint3_LREAL : HRESULT  
VAR_INPUT  
  ipContainer : ITCvNContainer;  
  aMax        : Reference To TcVnPoint3_LREAL;  
  hrPrev      : HRESULT;  
END_VAR

**Inputs**

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMax</td>
<td>Reference To TcVnPoint3_REAL [151]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.7.1.10 F_VN_MaxElementElementwise_TcVnPoint3_REAL

F_VN_MaxElementElementwise_TcVnPoint3_REAL

Gets the element wise maximum container element.

## Syntax

**Definition:**

FUNCTION F_VN_MaxElementElementwise_TcVnPoint3_REAL : HRESULT

VAR_INPUT

  ipContainer : ITcVnContainer;
  aMax        : Reference To TcVnPoint3_REAL;
  hrPrev      : HRESULT;

END_VAR
6.4.7.1.11  

**F_VN_MaxElementElementwise_TcVnVector2_DINT**

Gets the element wise maximum container element.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_MaxElementElementwise_TcVnVector2_DINT : HRESULT
VAR_INPUT
    ipContainer : ITCvnContainer;
    aMax        : Reference To TcVnVector2_DINT;
    hrPrev      : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvnContainer</td>
<td>Source container</td>
</tr>
<tr>
<td>aMax</td>
<td>Reference To TcVnVector2_DINT</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.7.1.12  F_VN_MaxElementElementwise_TcVnVector2_INT

F_VN_MaxElementElementwise_TcVnVector2_INT

ipContainer  ITCvnContainer  HRESULT
aMax  Reference To TcVnVector2_INT
hrPrev  HRESULT

Gets the element wise maximum container element.

Syntax

Definition:
FUNCTION F_VN_MaxElementElementwise_TcVnVector2_INT : HRESULT
VAR_INPUT
   ipContainer  : ITCvnContainer;
   aMax  : Reference To TcVnVector2_INT;
   hrPrev  : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMax</td>
<td>Reference To TcVnVector2_INT [153]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.1.13  F_VN_MaxElementElementwise_TcVnVector2_REAL

F_VN_MaxElementElementwise_TcVnVector2_REAL

ipContainer  ITCvnContainer  HRESULT
aMax  Reference To TcVnVector2_REAL
hrPrev  HRESULT

Gets the element wise maximum container element.

Syntax

Definition:
FUNCTION F_VN_MaxElementElementwise_TcVnVector2_REAL : HRESULT
VAR_INPUT
   ipContainer  : ITCvnContainer;
   aMax  : Reference To TcVnVector2_REAL;
   hrPrev  : HRESULT;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Source container</td>
</tr>
<tr>
<td>aMax</td>
<td>Reference To TcVnVector2_SINT</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## Syntax

### Definition:

FUNCTION F_VN_MaxElementElementwise_TcVnVector2_SINT : HRESULT

VAR_INPUT

  ipContainer : ITcVnContainer;
  aMax        : Reference To TcVnVector2_SINT;
  hrPrev      : HRESULT;

END_VAR

Gets the element wise maximum container element.

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Source container</td>
</tr>
<tr>
<td>aMax</td>
<td>Reference To TcVnVector2_SINT</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]
Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.15  F_VN_MaxElementElementwise_TcVnVector2_UINT

Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_TcVnVector2_UINT : HRESULT
VAR_INPUT
    ipContainer : ITCvNContainer;
    aMax        : Reference To TcVnVector2_UINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMax</td>
<td>Reference To TcVnVector2_UINT [153]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.7.1.16  **F_VN_MaxElementElementwise_TcVnVector2_USINT**

**Syntax**

Definition:

```
FUNCTION F_VN_MaxElementElementwise_TcVnVector2_USINT : HRESULT
VAR_INPUT
    ipContainer : ITCvContainer;
    aMax : Reference To TcVnVector2_USINT;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMax</td>
<td>Reference To TcVnVector2_USINT [153]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
</table>
| TwinCAT V3.1.4024.17 or later | PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU | Tc3_Vision

6.4.7.1.17  **F_VN_MaxElementElementwise_TcVnVector3_INT**

**Syntax**

Definition:

```
FUNCTION F_VN_MaxElementElementwise_TcVnVector3_INT : HRESULT
VAR_INPUT
    ipContainer : ITCvContainer;
```

Gets the element wise maximum container element.

**Inputs**
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer } 345</td>
<td>Source container</td>
</tr>
<tr>
<td>aMax</td>
<td>Reference To TcVnVector3_INT } 153</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT } 135</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT } 135

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.18 F_VN_MaxElementElementwise_TcVnVector3_REAL

Gets the element wise maximum container element.

Syntax

Definition:

FUNCTION F_VN_MaxElementElementwise_TcVnVector3_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aMax       : Reference To TcVnVector3_REAL;
    hrPrev     : HRESULT;
END_VAR
Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.1.19 F_VN_MaxElementElementwise_TcVnVector3_SINT

```c
F_VN_MaxElementElementwise_TcVnVector3_SINT
```

Gets the element wise maximum container element.

Syntax

Definition:

```c
FUNCTION F_VN_MaxElementElementwise_TcVnVector3_SINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aMax : Reference To TcVnVector3_SINT;
    hrPrev : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMax</td>
<td>Reference To TcVnVector3_SINT [153]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
# 6.4.7.1.20 F_VN_MaxElementElementwise_TcVnVector3_UINT

## Gets the element wise maximum container element.

### Syntax

**Definition:**

```
FUNCTION F_VN_MaxElementElementwise_TcVnVector3_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aMax        : Reference To TcVnVector3_UINT;
    hrPrev      : HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMax</td>
<td>Reference To TcVnVector3_UINT [153]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

**HRESULT [135]**

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

---

## 6.4.7.1.21 F_VN_MaxElementElementwise_TcVnVector3_USINT

## Gets the element wise maximum container element.

### Syntax

**Definition:**

```
FUNCTION F_VN_MaxElementElementwise_TcVnVector3_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aMax        : Reference To TcVnVector3_USINT;
    hrPrev      : HRESULT;
END_VAR
```
# Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Source container</td>
</tr>
<tr>
<td>aMax</td>
<td>Reference To TcVnVector3_USINT</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.7.1.22 F_VN_MaxElementElementwise_TcVnVector4_DINT

Gets the element wise maximum container element.

### Syntax

**Definition:**

```c
FUNCTION F_VN_MaxElementElementwise_TcVnVector4_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aMax : Reference To TcVnVector4_DINT;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Source container</td>
</tr>
<tr>
<td>aMax</td>
<td>Reference To TcVnVector4_DINT [153]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]
Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.1.23  F_VN_MaxElementElementwise_TcVnVector4_INT

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMax</td>
<td>Reference To TcVnVector4_INT [153]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.7.1.24 F_VN_MaxElementElementwise_TcVnVector4_LREAL

 Gets the element wise maximum container element.

 Syntax

 Definition:

 ```c
 FUNCTION F_VN_MaxElementElementwise_TcVnVector4_LREAL : HRESULT
 VAR_INPUT
   ipContainer : ITCvNContainer;
   aMax        : Reference To TcVnVector4_LREAL;
   hrPrev      : HRESULT;
 END_VAR
 ```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer</td>
<td>Source container</td>
</tr>
<tr>
<td>aMax</td>
<td>Reference To</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td></td>
<td>TcVnVector4_LREAL</td>
<td></td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

 Return value

 HRESULT [135]

 Required License

 TC3 Vision Base

 System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

 6.4.7.1.25 F_VN_MaxElementElementwise_TcVnVector4_SINT

 Gets the element wise maximum container element.

 Syntax

 Definition:

 ```c
 FUNCTION F_VN_MaxElementElementwise_TcVnVector4_SINT : HRESULT
 VAR_INPUT
   ipContainer : ITCvNContainer;
 ```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMax</td>
<td>Reference To TcVnVector4_UINT [153]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TWINCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.1.26  F_VN_MaxElementElementwise_TcVnVector4_UINT

Gets the element wise maximum container element.

**Syntax**

**Definition:**

FUNCTION F_VN_MaxElementElementwise_TcVnVector4_UINT : HRESULT

VAR_INPUT

    ipContainer : ITcVnContainer;
    aMax : Reference To TcVnVector4_UINT;
    hrPrev : HRESULT;

END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMax</td>
<td>Reference To TcVnVector4_UINT [153]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.1.27  F_VN_MaxElementElementwise_TcVnVector4_USINT

F_VN_MaxElementElementwise_TcVnVector4_USINT

ipContainer : ITCvNContainer
hrPrev : HRESULT

aMax : Reference To TcVnVector4_USINT

Gets the element wise maximum container element.

Syntax

Definition:

FUNCTION F_VN_MaxElementElementwise_TcVnVector4_USINT : HRESULT
VARIABLE_INPUT
    ipContainer : ITCvNContainer;
    aMax : Reference To TcVnVector4_USINT;
    hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMax</td>
<td>Reference To TcVnVector4_USINT [153]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
### 6.4.7.1.28 F_VN_MaxElementElementwise_UDINT

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>F_VN_MaxElementElementwise_UDINT</code></td>
<td>Gets the element wise maximum container element.</td>
</tr>
</tbody>
</table>

#### Syntax

**Definition:**

```c
FUNCTION F_VN_MaxElementElementwise_UDINT : HRESULT
VAR_INPUT
    ipContainer : ITCvNContainer;
    nMax        : Reference To UDINT;
    hrPrev      : HRESULT;
END_VAR
```

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ipContainer</code></td>
<td><code>ITcVnContainer</code></td>
<td>Source container</td>
</tr>
<tr>
<td><code>nMax</code></td>
<td><code>Reference To UDINT</code></td>
<td>Returns the requested element</td>
</tr>
<tr>
<td><code>hrPrev</code></td>
<td><code>HRESULT</code></td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

#### Return Value

`HRESULT` [135]

#### Required License

TC3 Vision Base

#### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

---

### 6.4.7.1.29 F_VN_MaxElementElementwise_UINT

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>F_VN_MaxElementElementwise_UINT</code></td>
<td>Gets the element wise maximum container element.</td>
</tr>
</tbody>
</table>

#### Syntax

**Definition:**

```c
FUNCTION F_VN_MaxElementElementwise_UINT : HRESULT
VAR_INPUT
    ipContainer : ITCvNContainer;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>nMax</td>
<td>Reference To UINT</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.7.1.30 F_VN_MaxElementElementwise_ULINT**

```plaintext
definition:
FUNCTION F_VN_MaxElementElementwise_ULINT : HRESULT
VAR_INPUT
  ipContainer : ITcVnContainer;
  nMax : Reference To UINT;
  hrPrev : HRESULT;
END_VAR
```

Gets the element wise maximum container element.

**Syntax**

**Definition:**

FUNCTION F_VN_MaxElementElementwise_ULINT : HRESULT
VAR_INPUT
  ipContainer : ITcVnContainer;
  nMax : Reference To UINT;
  hrPrev : HRESULT;
END_VAR
Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.1.31  F_VN_MaxElementElementwise_USINT

**F_VN_MaxElementElementwise_USINT**

- `ipContainer` : `ITcVnContainer`
- `nMax` : `Reference To USINT`
- `hrPrev` : `HRESULT`

Gets the element wise maximum container element.

**Syntax**

Definition:

```c
FUNCTION F_VN_MaxElementElementwise_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nMax : Reference To USINT;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ipContainer</code></td>
<td><code>ITcVnContainer</code> ![345]</td>
<td>Source container</td>
</tr>
<tr>
<td><code>nMax</code></td>
<td><code>Reference To USINT</code></td>
<td>Returns the requested element</td>
</tr>
<tr>
<td><code>hrPrev</code></td>
<td><code>HRESULT</code> ![135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

`HRESULT` ![135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.7.2 F_VN_MedianElementElementwise

6.4.7.2.1 F_VN_MedianElementElementwise_DINT

Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_DINT : HRESULT
VAR_INPUT
   ipContainer : ITcVnContainer;
   nMedian     : Reference To DINT;
   hrPrev      : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Source container</td>
</tr>
<tr>
<td>nMedian</td>
<td>Reference To DINT</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.2.2 F_VN_MedianElementElementwise_INT

Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_INT : HRESULT
VAR_INPUT
   ipContainer : ITcVnContainer;
   nMedian     : Reference To INT;
   hrPrev      : HRESULT;
END_VAR
```

This content is a detailed explanation of the CTP vision functions for calculating the median of an element-wise container. It includes the function definitions, input parameters, and system requirements. The API reference provides essential information for developers who need to integrate these functions into their applications.
FUNCTION F_VN_MedianElementElementwise_INT : HRESULT
VAR_INPUT
   ipContainer : ITcVnContainer;
   nMedian     : Reference To INT;
   hrPrev      : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>nMedian</td>
<td>Reference To INT</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.2.3 F_VN_MedianElementElementwise_LREAL

F_VN_MedianElementElementwise_LREAL

<table>
<thead>
<tr>
<th>ipContainer</th>
<th>ITcVnContainer</th>
</tr>
</thead>
<tbody>
<tr>
<td>fMedian</td>
<td>Reference To LREAL</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
</tr>
</tbody>
</table>

Gets the element wise median container element.

Syntax

Definition:

FUNCTION F_VN_MedianElementElementwise_LREAL : HRESULT
VAR_INPUT
   ipContainer : ITcVnContainer;
   fMedian     : Reference To LREAL;
   hrPrev      : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>fMedian</td>
<td>Reference To LREAL</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.7.2.4 F_VN_MedianElementElementwise_REAL

Gets the element wise median container element.

**Syntax**

**Definition:**

FUNCTION F_VN_MedianElementElementwise_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    fMedian     : Reference To REAL;
    hrPrev      : HRESULT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>fMedian</td>
<td>Reference To REAL</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.7.2.5  F_VN_MedianElementElementwise_SINT

Gets the element wise median container element.

Syntax

Definition:

FUNCTION F_VN_MedianElementElementwise_SINT : HRESULT
VAR_INPUT
    ipContainer : ITCvNContainer;
    nMedian : Reference To SINT;
    hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>nMedian</td>
<td>Reference To SINT</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.2.6  F_VN_MedianElementElementwise_TcVnPoint2_DINT

Gets the element wise median container element.

Syntax

Definition:

FUNCTION F_VN_MedianElementElementwise_TcVnPoint2_DINT : HRESULT
VAR_INPUT
    ipContainer : ITCvNContainer;
    aMedian : Reference To TcVnPoint2_DINT;
    hrPrev : HRESULT;
END_VAR
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMedian</td>
<td>Reference To TcVnPoin t2_LREAL [151]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.2.7 **F_VN_MedianElementElementwise_TcVnPoin t2_LREAL**

Gets the element wise median container element.

**Syntax**

Definition:

FUNCTION F_VN_MedianElementElementwise_TcVnPoin t2_LREAL : HRESULT
VAR_INPUT
  ipContainer : ITcVnContainer;
  aMedian : Reference To TcVnPoin t2_LREAL;
  hrPrev : HRESULT;
END_VAR

**Return value**

HRESULT [135]
6.4.7.2.8  F_VN_MedianElementElementwise_TcVnPoint2_REAL

 Gets the element wise median container element.

 Syntax

 **Definition:**

 ```
 FUNCTION F_VN_MedianElementElementwise_TcVnPoint2_REAL : HRESULT
 VAR_INPUT
 ipContainer : ITCvnContainer;
 aMedian     : Reference To TcVnPoint2_REAL;
 hrPrev      : HRESULT;
 END_VAR
 ```

 **Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvnContainer</td>
<td>Source container</td>
</tr>
<tr>
<td>aMedian</td>
<td>Reference To TcVnPoint2_REAL</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

 **Return value**

 HRESULT [135]

 Required License

 TC3 Vision Base

 System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

 6.4.7.2.9  F_VN_MedianElementElementwise_TcVnPoint3_LREAL


Gets the element wise median container element.

**Syntax**

**Definition:**

```
FUNCTION F_VN_MedianElementElementwise_TcVnPoint3_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aMedian      : Reference To TcVnPoint3_LREAL;
    hrPrev       : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMedian</td>
<td>Reference To TcVnPoint3_REAL [151]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TWINCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.2.10  **F_VN_MedianElementElementwise_TcVnPoint3_REAL**

```
F_VN_MedianElementElementwise_TcVnPoint3_REAL
ipContainer ITcVnContainer
aMedian Reference To TcVnPoint3_REAL
hrPrev HRESULT
```

Gets the element wise median container element.

**Syntax**

**Definition:**

```
FUNCTION F_VN_MedianElementElementwise_TcVnPoint3_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aMedian      : Reference To TcVnPoint3_REAL;
    hrPrev       : HRESULT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMedian</td>
<td>Reference To TcVnPoint3 REAL [151]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.7.2.11 F_VN_MedianElementElementwise_TcVnVector2_DINT**

Gets the element wise median container element.

**Syntax**

**Definition:**

FUNCTION F_VN_MedianElementElementwise_TcVnVector2_DINT : HRESULT

VAR_INPUT

  ipContainer : ITcVnContainer;
  aMedian : Reference To TcVnVector2_DINT;
  hrPrev : HRESULT;

END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMedian</td>
<td>Reference To TcVnVector2_DINT [153]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]
6.4.7.2.12  F_VN_MedianElementElementwise_TcVnVector2_INT

F_VN_MedianElementElementwise_TcVnVector2_INT

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvVnContainer</td>
<td>Source container</td>
</tr>
<tr>
<td>aMedian</td>
<td>Reference To TcVnVector2_INT</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.2.13  F_VN_MedianElementElementwise_TcVnVector2_REAL

F_VN_MedianElementElementwise_TcVnVector2_REAL

Gets the element wise median container element.

**Syntax**

**Definition:**

FUNCTION F_VN_MedianElementElementwise_TcVnVector2_INT : HRESULT

VAR_INPUT

ipContainer : ITCvVnContainer;
aMedian    : Reference To TcVnVector2_INT;
hrPrev     : HRESULT;

END_VAR
Gets the element wise median container element.

**Syntax**

**Definition:**

FUNCTION F_VN_MedianElementElementwise_TcVnVector2_REAL : HRESULT

VAR_INPUT

    ipContainer : ITcVnContainer;
    aMedian     : Reference To TcVnVector2_REAL;
    hrPrev      : HRESULT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMedian</td>
<td>Reference To TcVnVector2_REAL [153]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.2.14 F_VN_MedianElementElementwise_TcVnVector2_SINT

Gets the element wise median container element.

**Syntax**

**Definition:**

FUNCTION F_VN_MedianElementElementwise_TcVnVector2_SINT : HRESULT

VAR_INPUT

    ipContainer : ITcVnContainer;
    aMedian     : Reference To TcVnVector2_SINT;
    hrPrev      : HRESULT;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Source container</td>
</tr>
<tr>
<td>aMedian</td>
<td>Reference To TcVnVector2_UINT</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.7.2.15 F_VN_MedianElementElementwise_TcVnVector2_UINT

Gets the element wise median container element.

## Syntax

**Definition:**

```plaintext
FUNCTION F_VN_MedianElementElementwise_TcVnVector2_UINT : HRESULT  
VAR_INPUT  
ipContainer : ITcVnContainer;  
aMedian : Reference To TcVnVector2_UINT;  
hrPrev : HRESULT;  
END_VAR
```
API reference

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.2.16 F_VN_MedianElementElementwise_TcVnVector2_USINT

Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_TcVnVector2_USINT : HRESULT
VAR_INPUT
    ipContainer : ITCvNContainer;
    aMedian     : Reference To TcVnVector2_USINT;
    hrPrev      : HRESULT;
END_VAR
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer[ ] 345</td>
<td>Source container</td>
</tr>
<tr>
<td>aMedian</td>
<td>Reference To TcVnVector2_USINT[ ] 153</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[ ] 135</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.7.2.17  F_VN_MedianElementElementwise_TcVnVector3_INT

Gets the element wise median container element.

**Syntax**

Definition:

FUNCTION F_VN_MedianElementElementwise_TcVnVector3_INT : HRESULT
VAR_INPUT
   ipContainer : ITCvNContainer;
   aMedian     : Reference To TcVnVector3_INT;
   hrPrev      : HRESULT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMedian</td>
<td>Reference To TcVnVector3_INT [153]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

---

6.4.7.2.18  F_VN_MedianElementElementwise_TcVnVector3_REAL

Gets the element wise median container element.

**Syntax**

Definition:

FUNCTION F_VN_MedianElementElementwise_TcVnVector3_REAL : HRESULT
VAR_INPUT
   ipContainer : ITCvNContainer;
   aMedian     : Reference To TcVnVector3_REAL;
   hrPrev      : HRESULT;
END_VAR
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [ITcVnContainer [345]}</td>
<td>Source container</td>
</tr>
<tr>
<td>aMedian</td>
<td>Reference To TcVnVector3_REAL [153]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [HRESULT [135]}</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.2.19  F_VN_MedianElementElementwise_TcVnVector3_SINT

Gets the element wise median container element.

**Syntax**

**Definition:**

FUNCTION F_VN_MedianElementElementwise_TcVnVector3_SINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aMedian     : Reference To TcVnVector3_SINT;
    hrPrev      : HRESULT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [ITcVnContainer [345]}</td>
<td>Source container</td>
</tr>
<tr>
<td>aMedian</td>
<td>Reference To TcVnVector3_SINT [153]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [HRESULT [135]}</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]
6.4.7.20  F_VN_MedianElementElementwise_TcVnVector3_UINT

Gets the element wise median container element.

Syntax

Definition:
FUNCTION F_VN_MedianElementElementwise_TcVnVector3_UINT : HRESULT
VAR_INPUT
  ipContainer : ITcVnContainer;
  aMedian      : Reference To TcVnVector3_UINT;
  hrPrev      : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMedian</td>
<td>Reference To TcVnVector3_UINT [153]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g.</td>
<td>Tc3_Vision</td>
</tr>
<tr>
<td></td>
<td>Intel 4-core Atom CPU</td>
<td></td>
</tr>
</tbody>
</table>
Gets the element wise median container element.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_MedianElementElementwise_TcVnVector3_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aMedian      : Reference To TcVnVector3_USINT;
    hrPrev       : HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Source container</td>
</tr>
<tr>
<td>aMedian</td>
<td>Reference To TcVnVector3_USINT</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.22  F_VN_MedianElementElementwise_TcVnVector4_DINT

```c
F_VN_MedianElementElementwise_TcVnVector4_DINT
    ipContainer : ITcVnContainer;
    aMedian      : Reference To TcVnVector4_DINT;
    hrPrev       : HRESULT;
```

Gets the element wise median container element.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_MedianElementElementwise_TcVnVector4_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aMedian      : Reference To TcVnVector4_DINT;
    hrPrev       : HRESULT;
END_VAR
```
# Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Source container</td>
</tr>
<tr>
<td>aMedian</td>
<td>Reference To TcVnVector4_INT</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

# Return value

HRESULT [135]

# Required License

TC3 Vision Base

# System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.7.2.23 F_VN_MedianElementElementwise_TcVnVector4_INT

Gets the element wise median container element.

### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_MedianElementElementwise_TcVnVector4_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aMedian     : Reference To TcVnVector4_INT;
    hrPrev      : HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Source container</td>
</tr>
<tr>
<td>aMedian</td>
<td>Reference To TcVnVector4_INT</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]
Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.2.24  F_VN_MedianElementElementwise_TcVnVector4_LREAL

GETS THE ELEMENT WISE MEDIAN CONTAINER ELEMENT.

Syntax

Definition:

FUNCTION F_VN_MedianElementElementwise_TcVnVector4_LREAL : HRESULT
VAR_INPUT
  ipContainer : ITcVnContainer;
  aMedian : Reference To TcVnVector4_LREAL;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [1845]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMedian</td>
<td>Reference To TcVnVector4_LREAL [1953]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [1835]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [1835]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.7.2.25  F_VN_MedianElementElementwise_TcVnVector4_SINT

Gets the element wise median container element.

Syntax

Definition:

FUNCTION F_VN_MedianElementElementwise_TcVnVector4_SINT : HRESULT
VAR_INPUT
   ipContainer : ITCvnContainer;
   aMedian : Reference To TcVnVector4_SINT;
   hrPrev : HRESULT;
END_VAR

= Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMedian</td>
<td>Reference To TcVnVector4_SINT [153]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

= Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.2.26  F_VN_MedianElementElementwise_TcVnVector4_UINT

Gets the element wise median container element.

Syntax

Definition:

FUNCTION F_VN_MedianElementElementwise_TcVnVector4_UINT : HRESULT
VAR_INPUT
   ipContainer : ITCvnContainer;
   aMedian : Reference To TcVnVector4_UINT;
   hrPrev : HRESULT;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMedian</td>
<td>Reference To TcVnVector4_UINT [153]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

### Development environment

TwinCAT V3.1.4024.17 or later

### Target platform

PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU

### PLC libraries to include

Tc3_Vision

### 6.4.7.2.27 F_VN_MedianElementElementwise_TcVnVector4_USINT

Gets the element wise median container element.

**Syntax**

**Definition:**

FUNCTION F_VN_MedianElementElementwise_TcVnVector4_USINT : HRESULT

VAR_INPUT

ipContainer : ITcVnContainer;
aMedian : Reference To TcVnVector4_UINT;
hrPrev : HRESULT;

END_VAR

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMedian</td>
<td>Reference To TcVnVector4_UINT [153]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]
Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.2.28  F_VN_MedianElementElementwise_UDINT

gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_UDINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nMedian : Reference To UDINT;
    hrPrev : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>nMedian</td>
<td>Reference To UDINT</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.7.2.29  F_VN_MedianElementElementwise_UINT

Gets the element wise median container element.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_MedianElementElementwise_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nMedian : Reference To UINT;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Source container</td>
</tr>
<tr>
<td>nMedian</td>
<td>Reference To UINT</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.2.30  F_VN_MedianElementElementwise_ULINT

Gets the element wise median container element.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_MedianElementElementwise_ULINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>nMedian</td>
<td>Reference To USINT</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT[135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.7.2.31 F_VN_MedianElementElementwise_USINT

F_VN_MedianElementElementwise_USINT

ipContainer ITcVnContainer

nMedian Reference To USINT

hrPrev HRESULT

F_VN_MedianElementElementwise_USINT

Gets the element wise median container element.

## Syntax

**Definition:**

FUNCTION F_VN_MedianElementElementwise_USINT : HRESULT

VAR_INPUT

- ipContainer : ITcVnContainer;
- nMedian : Reference To USINT;
- hrPrev : HRESULT;

END_VAR

## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>nMedian</td>
<td>Reference To USINT</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT[135]
6.4.7.3 F_VN_MinElementElementwise

6.4.7.3.1 F_VN_MinElementElementwise_DINT

F_VN_MinElementElementwise_DINT

Gets the element wise minimum container element.

Syntax

Definition:

```c
FUNCTION F_VN_MinElementElementwise_DINT : HRESULT
VAR_INPUT
   ipContainer : ITCvNContainer;
   nMin        : Reference To DINT;
   hrPrev      : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>nMin</td>
<td>Reference To DINT</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.7.3.2 **F_VN_MinElementElementwise_INT**

```plaintext
FUNCTION F_VN_MinElementElementwise_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nMin        : Reference To INT;
    hrPrev      : HRESULT;
END_VAR
```

Gets the element wise minimum container element.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_MinElementElementwise_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nMin        : Reference To INT;
    hrPrev      : HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Source container</td>
</tr>
<tr>
<td>nMin</td>
<td>Reference To INT</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

**HRESULT [135]**

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.3.3 **F_VN_MinElementElementwise_LREAL**

```plaintext
FUNCTION F_VN_MinElementElementwise_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    fMin        : Reference To LREAL;
    hrPrev      : HRESULT;
END_VAR
```

Gets the element wise minimum container element.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_MinElementElementwise_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>fMin</td>
<td>Reference To REAL</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.3.4  

F_VN_MinElementElementwise_REAL

F_VN_MinElementElementwise_REAL

ipContainer : ITcVnContainer

fMin : Reference To REAL

hrPrev : HRESULT

GETS THE ELEMENT WISE MINIMUM CONTAINER ELEMENT.

**Syntax**

**Definition:**

FUNCTION F_VN_MinElementElementwise_REAL : HRESULT

VAR_INPUT

ipContainer : ITcVnContainer;

fMin : Reference To REAL;

hrPrev : HRESULT;

END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>fMin</td>
<td>Reference To REAL</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
### Return value

**HRESULT [135]**

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.7.3.5 **F_VN_MinElementElementwise_SINT**

**F_VN_MinElementElementwise_SINT**

- **ipContainer** : ITC3VnContainer
- **nMin** : Reference To SINT
- **hrPrev** : HRESULT

Gets the element wise minimum container element.

### Syntax

**Definition**:

```plaintext
FUNCTION F_VN_MinElementElementwise_SINT : HRESULT
VAR_INPUT
    ipContainer : ITC3VnContainer;
    nMin : Reference To SINT;
    hrPrev : HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITC3VnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>nMin</td>
<td>Reference To SINT</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

**HRESULT [135]**

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.7.3.6  F_VN_MinElementElementwise_TcVnPoint2_DINT

F_VN_MinElementElementwise_TcVnPoint2_DINT

ipContainer  : ITcVnContainer
aMin  : Reference To TcVnPoint2_DINT
hrPrev  : HRESULT

Gets the element wise minimum container element.

Syntax

Definition:
FUNCTION F_VN_MinElementElementwise_TcVnPoint2_DINT : HRESULT
VAR_INPUT
    ipContainer  : ITcVnContainer;
    aMin  : Reference To TcVnPoint2_DINT;
    hrPrev  : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMin</td>
<td>Reference To TcVnPoint2_DINT</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.3.7  F_VN_MinElementElementwise_TcVnPoint2_LREAL

F_VN_MinElementElementwise_TcVnPoint2_LREAL

ipContainer  : ITcVnContainer
aMin  : Reference To TcVnPoint2_LREAL
hrPrev  : HRESULT

Gets the element wise minimum container element.

Syntax

Definition:
FUNCTION F_VN_MinElementElementwise_TcVnPoint2_LREAL : HRESULT
VAR_INPUT
    ipContainer  : ITcVnContainer;
    aMin  : Reference To TcVnPoint2_LREAL;
    hrPrev  : HRESULT;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMin</td>
<td>Reference To TcVnPoin2_REAL [151]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.7.3.8 F_VN_MinElementElementwise_TcVnPoin2_REAL

<table>
<thead>
<tr>
<th>Function: F_VN_MinElementElementwise_TcVnPoin2_REAL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer: ITcVnContainer</td>
<td></td>
</tr>
<tr>
<td>aMin: Reference To TcVnPoin2_REAL</td>
<td></td>
</tr>
<tr>
<td>hrPrev: HRESULT</td>
<td></td>
</tr>
</tbody>
</table>

Gets the element wise minimum container element.

### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_MinElementElementwise_TcVnPoin2_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aMin : Reference To TcVnPoin2_REAL;
    hrPrev : HRESULT;
END_VAR
```

## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMin</td>
<td>Reference To TcVnPoin2_REAL [151]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]
6.4.7.3.9  F_VN_MinElementElementwise_TcVnPoint3_LREAL

Gets the element wise minimum container element.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_MinElementElementwise_TcVnPoint3_LREAL : HRESULT
VAR_INPUT
   ipContainer : ITCvContainer;
   aMin        : Reference To TcVnPoint3_LREAL;
   hrPrev      : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvContainer</td>
<td>Source container</td>
</tr>
<tr>
<td>aMin</td>
<td>Reference To TcVnPoint3_LREAL</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.7.3.10  F_VN_MinElementElementwise_TcVnPoint3_REAL

Gets the element wise minimum container element.

**Syntax**

**Definition:**

```
FUNCTION F_VN_MinElementElementwise_TcVnPoint3_REAL : HRESULT
VAR_INPUT
  ipContainer : ITCvVnContainer;
  aMin        : Reference To TcVnPoint3_REAL;
  hrPrev      : HRESULT;
END_VAR
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvVnContainer</td>
<td>Source container</td>
</tr>
<tr>
<td>aMin</td>
<td>Reference To TcVnPoint3_REAL</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.3.11  F_VN_MinElementElementwise_TcVnVector2_DINT

Gets the element wise minimum container element.

**Syntax**

**Definition:**

```
FUNCTION F_VN_MinElementElementwise_TcVnVector2_DINT : HRESULT
VAR_INPUT
  ipContainer : ITCvVnContainer;
  aMin        : Reference To TcVnVector2_DINT;
  hrPrev      : HRESULT;
END_VAR
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvVnContainer</td>
<td>Source container</td>
</tr>
<tr>
<td>aMin</td>
<td>Reference To TcVnVector2_DINT</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMin</td>
<td>Reference To TcVnVector2_INT [153]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.3.12 **F_VN_MinElementElementwise_TcVnVector2_INT**

```
FUNCTION F_VN_MinElementElementwise_TcVnVector2_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aMin : Reference To TcVnVector2_INT;
    hrPrev : HRESULT;
END_VAR
```

Gets the element wise minimum container element.

**Syntax**

**Definition:**

FUNCTION F_VN_MinElementElementwise_TcVnVector2_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aMin : Reference To TcVnVector2_INT;
    hrPrev : HRESULT;
END_VAR
## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.7.3.13 F_VN_MinElementElementwise_TcVnVector2_REAL

![Function Diagram](image)

Gets the element wise minimum container element.

#### Syntax

**Definition:**

```c
FUNCTION F_VN_MinElementElementwise_TcVnVector2_REAL : HRESULT
VAR_INPUT
    ipContainer : ITCvnContainer;
    aMin        : Reference To TcVnVector2_REAL;
    hrPrev      : HRESULT;
END_VAR
```

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvnContainer ![345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMin</td>
<td>Reference To TcVnVector2_REAL ![153]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT ![135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

#### Return value

**HRESULT ![135]**

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.7.3.14 F_VN_MinElementElementwise_TcVnVector2_SINT

Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_TcVnVector2_SINT : HRESULT
VAR_INPUT
  ipContainer : ITCvncContainer;
  aMin : Reference To TcVnVector2_SINT;
  hrPrev : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvncContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMin</td>
<td>Reference To TcVnVector2_SINT [153]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.3.15 F_VN_MinElementElementwise_TcVnVector2(UINT)

Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_TcVnVector2_UINT : HRESULT
VAR_INPUT
  ipContainer : ITCvncContainer;
  aMin : Reference To TcVnVector2_UINT;
  hrPrev : HRESULT;
END_VAR
```
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMin</td>
<td>Reference To TcVnVector2_UINT [153]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.3.16 F_VN_MinElementElementwise_TcVnVector2_USINT

 Gets the element wise minimum container element.

Syntax

Definition:

FUNCTION F_VN_MinElementElementwise_TcVnVector2_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aMin : Reference To TcVnVector2_UINT;
    hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMin</td>
<td>Reference To TcVnVector2_UINT [153]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]
Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.3.17 F_VN_MinElementElementwise_TcVnVector3_INT

Gets the element wise minimum container element.

**Syntax**

**Definition:**

FUNCTION F_VN_MinElementElementwise_TcVnVector3_INT : HRESULT
VAR_INPUT
  ipContainer : ITCvNContainer;
  aMin : Reference To TcVnVector3_INT;
  hrPrev : HRESULT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMin</td>
<td>Reference To TcVnVector3_INT [153]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
### 6.4.7.3.18 F_VN_MinElementElementwise_TcVnVector3_REAL

**Definition:**

FUNCTION F_VN_MinElementElementwise_TcVnVector3_REAL : HRESULT

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMin</td>
<td>Reference To TcVnVector3_REAL [153]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.7.3.19 F_VN_MinElementElementwise_TcVnVector3_SINT

**Definition:**

FUNCTION F_VN_MinElementElementwise_TcVnVector3_SINT : HRESULT

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td></td>
</tr>
<tr>
<td>aMin</td>
<td>Reference To TcVnVector3_SINT</td>
<td></td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td></td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td><code>ITcVnContainer</code></td>
<td>Source container</td>
</tr>
<tr>
<td>aMin</td>
<td>Reference To <code>TcVnVector3_UINT</code></td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td><code>HRESULT</code></td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

`HRESULT` [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.7.3.20 `F_VN_MinElementElementwise_TcVnVector3_UINT`

Gets the element wise minimum container element.

#### Syntax

**Definition:**

```c
FUNCTION F_VN_MinElementElementwise_TcVnVector3_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aMin        : Reference To TcVnVector3_UINT;
    hrPrev      : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td><code>ITcVnContainer</code></td>
<td>Source container</td>
</tr>
<tr>
<td>aMin</td>
<td>Reference To <code>TcVnVector3_UINT</code></td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td><code>HRESULT</code></td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

`HRESULT` [135]
Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.7.3.21 F_VN_MinElementElementwise_TcVnVector3_USINT

**F_VN_MinElementElementwise_TcVnVector3_USINT**

- **ipContainer**: ITcVnContainer
- **aMin**: Reference To TcVnVector3_USINT
- **hrPrev**: HRESULT

Gets the element wise minimum container element.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_MinElementElementwise_TcVnVector3_USINT : HRESULT
VAR_INPUT
  ipContainer : ITcVnContainer;
  aMin        : Reference To TcVnVector3_USINT;
  hrPrev      : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Source container</td>
</tr>
<tr>
<td>aMin</td>
<td>Reference To TcVnVector3_USINT</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

<table>
<thead>
<tr>
<th>HRESULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>[135]</td>
</tr>
</tbody>
</table>

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.7.3.22 F_VN_MinElementElementwise_TcVnVector4_DINT

Gets the element wise minimum container element.

Syntax

Definition:

FUNCTION F_VN_MinElementElementwise_TcVnVector4_DINT : HRESULT
VAR_INPUT
   ipContainer : ITcVnContainer;
   aMin : Reference To TcVnVector4_DINT;
   hrPrev : HRESULT;
END_VAR

![Inputs](image)

- **Name**
  - ipContainer: ITcVnContainer
  - aMin: Reference To TcVnVector4_DINT
  - hrPrev: HRESULT

- **Type**
  - ITcVnContainer [345]
  - Reference To TcVnVector4_DINT [153]
  - HRESULT [135]

- **Description**
  - Source container
  - Returns the requested element
  - HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.3.23 F_VN_MinElementElementwise_TcVnVector4_INT

Gets the element wise minimum container element.

Syntax

Definition:

FUNCTION F_VN_MinElementElementwise_TcVnVector4_INT : HRESULT
VAR_INPUT
   ipContainer : ITcVnContainer;
   aMin : Reference To TcVnVector4_INT;
   hrPrev : HRESULT;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Source container</td>
</tr>
<tr>
<td>aMin</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### Syntax

Definition:

FUNCTION F_VN_MinElementElementwise_TcVnVector4_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    aMin : Reference To TcVnVector4_LREAL;
    hrPrev : HRESULT;
END_VAR

Gets the element wise minimum container element.
### Required License
TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.7.3.25 F_VN_MinElementElementwise_TcVnVector4_SINT

**F_VN_MinElementElementwise_TcVnVector4_SINT**

```
ipContainer : ITCvnContainer
aMin        : Reference To TcVnVector4_SINT;
hrPrev      : HRESULT;
```

Gets the element wise minimum container element.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_MinElementElementwise_TcVnVector4_SINT : HRESULT
VAR_INPUT
    ipContainer : ITCvnContainer;
    aMin        : Reference To TcVnVector4_SINT;
    hrPrev      : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvnContainer</td>
<td>Source container</td>
</tr>
<tr>
<td>aMin</td>
<td>Reference To TcVnVector4_SINT</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

 HRESULT

### Required License
TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.7.3.26  \textbf{F\_VN\_MinElementElementwise\_TcVnVector4\_UINT}

\begin{itemize}
\item \texttt{ipContainer} \textit{ITcVnContainer} \hspace{1cm} \texttt{hrPrev} \texttt{HRESULT}
\item \texttt{aMin} \texttt{Reference To TcVnVector4\_UINT}
\end{itemize}

Gets the element wise minimum container element.

\textbf{Syntax}

\textbf{Definition:}

\begin{verbatim}
FUNCTION F_VN_MinElementElementwise_TcVnVector4_UINT : HRESULT
VAR_INPUT
  ipContainer : ITcVnContainer;
  aMin : Reference To TcVnVector4_UINT;
  hrPrev : HRESULT;
END_VAR
\end{verbatim}

\textbf{Inputs}

\begin{itemize}
\item \textbf{Name} \hspace{1cm} \textbf{Type} \hspace{1cm} \textbf{Description}
\item \texttt{ipContainer} \texttt{ITcVnContainer} \cite{345} \hspace{1cm} Source container
\item \texttt{aMin} \texttt{Reference To TcVnVector4\_UINT} \cite{153} \hspace{1cm} Returns the requested element
\item \texttt{hrPrev} \texttt{HRESULT} \cite{135} \hspace{1cm} HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
\end{itemize}

\textbf{Return value}

\texttt{HRESULT} \cite{135}

\textbf{Required License}

TC3 Vision Base

\textbf{System Requirements}

\begin{itemize}
\item Development environment: TwinCAT V3.1.4024.17 or later
\item Target platform: PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU
\item PLC libraries to include: Tc3\_Vision
\end{itemize}

6.4.7.3.27  \textbf{F\_VN\_MinElementElementwise\_TcVnVector4\_USINT}

\begin{itemize}
\item \texttt{ipContainer} \textit{ITcVnContainer} \hspace{1cm} \texttt{hrPrev} \texttt{HRESULT}
\item \texttt{aMin} \texttt{Reference To TcVnVector4\_USINT}
\end{itemize}

Gets the element wise minimum container element.

\textbf{Syntax}

\textbf{Definition:}

\begin{verbatim}
FUNCTION F_VN_MinElementElementwise_TcVnVector4_USINT : HRESULT
VAR_INPUT
  ipContainer : ITcVnContainer;
  aMin : Reference To TcVnVector4_USINT;
  hrPrev : HRESULT;
END_VAR
\end{verbatim}
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>aMin</td>
<td>Reference To TcVnVector4_USINT [153]</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.3.28 F_VN_MinElementElementwise_UDINT

F_VN_MinElementElementwise_UDINT

ipContainer ITcVnContainer HRESULT F_VN_MinElementElementwise_UDINT

nMin Reference To UDINT

hrPrev HRESULT

Gets the element wise minimum container element.

Syntax

Definition:

FUNCTION F_VN_MinElementElementwise_UDINT : HRESULT
VAR_INPUT
  ipContainer : ITcVnContainer;
  nMin : Reference To UDINT;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>nMin</td>
<td>Reference To UDINT</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]
API reference

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.3.29  F_VN_MinElementElementwise_UINT

```
F_VN_MinElementElementwise_UINT

ipContainer ITCvNContainer HRESULT F_VN_MinElementElementwise_UINT
nMin Reference To UINT
hrPrev HRESULT

```

Gets the element wise minimum container element.

Syntax

Definition:

```FUNCTION F_VN_MinElementElementwise_UINT : HRESULT
VAR_INPUT
    ipContainer : ITCvNContainer;
    nMin        : Reference To UINT;
    hrPrev      : HRESULT;
END_VAR```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITCvNContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>nMin</td>
<td>Reference To UINT</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.7.3.30   F_VN_MinElementElementwise_ULINT

Gets the element wise minimum container element.

Syntax

Definition:

FUNCTION F_VN_MinElementElementwise_ULINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nMin        : Reference To ULINT;
    hrPrev      : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer</td>
<td>Source container</td>
</tr>
<tr>
<td>nMin</td>
<td>Reference To ULINT</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.3.31   F_VN_MinElementElementwise_USINT

Gets the element wise minimum container element.

Syntax

Definition:

FUNCTION F_VN_MinElementElementwise_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nMin        : Reference To USINT;
    hrPrev      : HRESULT;
END_VAR
API reference

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container</td>
</tr>
<tr>
<td>nMin</td>
<td>Reference To USINT</td>
<td>Returns the requested element</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.7.4 F_VN_ContainerAverage**

Calculate the average of elements in a container.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_ContainerAverage : HRESULT
VAR_INPUT
    ipSrcContainer : ITcVnContainer;
    fAverage : Reference To LREAL;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container (Arithmetic, point and vector element types are supported. For points and vectors, the euclidean norm is used.)</td>
</tr>
<tr>
<td>fAverage</td>
<td>Reference To LREAL</td>
<td>Returns the calculated average value</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Further information

The table below shows which functions can be used for which container type:

<table>
<thead>
<tr>
<th>Types of containers</th>
<th>Usable functions</th>
</tr>
</thead>
</table>
| Containers of elements | F_VN_ContainerAverage [840]  
                         | F_VN_ContainerAverageVariance [845] |
| Containers of vectors, each with 2 elements | F_VN_ContainerAverageElementwise2 [841]  
                                             | F_VN_ContainerAverageVarianceElementwise2 [846] |
| Containers of vectors, each with 3 elements | F_VN_ContainerAverageElementwise3 [842]  
                                               | F_VN_ContainerAverageVarianceElementwise3 [847] |
| Containers of vectors, each with 4 elements | F_VN_ContainerAverageElementwise4 [843]  
                                               | F_VN_ContainerAverageVarianceElementwise4 [848] |

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.5 F_VN_ContainerAverageElementwise2

Calculate the elementwise average of 2D elements in a container.

Syntax

Definition:

FUNCTION F_VN_ContainerAverageElementwise2 : HRESULT
VAR_INPUT
   ipSrcContainer : ITcVnContainer;
   aAverage : Reference To TcVnVector2_LREAL;
   hrPrev : HRESULT;
END_VAR
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcContainer</td>
<td>ITcVnContainer [134]</td>
<td>Source container with 2D points or vectors</td>
</tr>
<tr>
<td>aAverage</td>
<td>Reference To TcVnVector2 LREAL [153]</td>
<td>Returns the calculated average values</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Further information**

This function calculates the average values element by element for a container with two elementary sub-elements per container element.

The table below shows which functions can be used for which container type:

<table>
<thead>
<tr>
<th>Types of containers</th>
<th>Usable functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Containers of elements</td>
<td>F_VN_ContainerAverage [840]</td>
</tr>
<tr>
<td></td>
<td>F_VN_ContainerAverageVariance [845]</td>
</tr>
<tr>
<td>Containers of vectors, each with 2 elements</td>
<td>F_VN_ContainerAverageElementwise2 [841]</td>
</tr>
<tr>
<td></td>
<td>F_VN_ContainerAverageVarianceElementwise2 [846]</td>
</tr>
<tr>
<td>Containers of vectors, each with 3 elements</td>
<td>F_VN_ContainerAverageElementwise3 [842]</td>
</tr>
<tr>
<td></td>
<td>F_VN_ContainerAverageVarianceElementwise3 [847]</td>
</tr>
<tr>
<td>Containers of vectors, each with 4 elements</td>
<td>F_VN_ContainerAverageElementwise4 [843]</td>
</tr>
<tr>
<td></td>
<td>F_VN_ContainerAverageVarianceElementwise4 [848]</td>
</tr>
</tbody>
</table>

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.7.6 F_VN_ContainerAverageElementwise3**

F_VN_ContainerAverageElementwise3

Calculate the elementwise average of 3D elements in a container.

**Syntax**

Definition:
FUNCTION F_VN_ContainerAverageElementwise3 : HRESULT
VAR_INPUT
ipSrcContainer : ITcVnContainer;
aAverage       : Reference To TcVnVector3_LREAL;
hrPrev         : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container with 3D points or vectors</td>
</tr>
<tr>
<td>aAverage</td>
<td>Reference To TcVnVector3_LREAL [153]</td>
<td>Returns the calculated average values</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Further information

The table below shows which functions can be used for which container type:

<table>
<thead>
<tr>
<th>Types of containers</th>
<th>Usable functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Containers of elements</td>
<td>F_VN_ContainerAverage [840]</td>
</tr>
<tr>
<td></td>
<td>F_VN_ContainerAverageVariance [845]</td>
</tr>
<tr>
<td>Containers of vectors, each with 2 elements</td>
<td>F_VN_ContainerAverageElementwise2 [841]</td>
</tr>
<tr>
<td></td>
<td>F_VN_ContainerAverageVarianceElementwise2 [846]</td>
</tr>
<tr>
<td>Containers of vectors, each with 3 elements</td>
<td>F_VN_ContainerAverageElementwise3 [842]</td>
</tr>
<tr>
<td></td>
<td>F_VN_ContainerAverageVarianceElementwise3 [847]</td>
</tr>
<tr>
<td>Containers of vectors, each with 4 elements</td>
<td>F_VN_ContainerAverageElementwise4 [843]</td>
</tr>
<tr>
<td></td>
<td>F_VN_ContainerAverageVarianceElementwise4 [848]</td>
</tr>
</tbody>
</table>

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.7 F_VN_ContainerAverageElementwise4

Calculate the elementwise average of 4D elements in a container.
Syntax

Definition:

FUNCTION F_VN_ContainerAverageElementwise4 : HRESULT
VAR_INPUT
  ipSrcContainer : ITcVnContainer;
  aAverage       : Reference To TcVnVector4_LREAL;
  hrPrev         : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container with 4D vectors</td>
</tr>
<tr>
<td>aAverage</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Returns the calculated average values</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Further information

The table below shows which functions can be used for which container type:

<table>
<thead>
<tr>
<th>Types of containers</th>
<th>Usable functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Containers of elements</td>
<td>F_VN_ContainerAverage [840]</td>
</tr>
<tr>
<td></td>
<td>F_VN_ContainerAverageVariance [845]</td>
</tr>
<tr>
<td>Containers of vectors, each with 2 elements</td>
<td>F_VN_ContainerAverageElementwise2 [841]</td>
</tr>
<tr>
<td></td>
<td>F_VN_ContainerAverageVarianceElementwise2 [846]</td>
</tr>
<tr>
<td>Containers of vectors, each with 3 elements</td>
<td>F_VN_ContainerAverageElementwise3 [842]</td>
</tr>
<tr>
<td></td>
<td>F_VN_ContainerAverageVarianceElementwise3 [847]</td>
</tr>
<tr>
<td>Containers of vectors, each with 4 elements</td>
<td>F_VN_ContainerAverageElementwise4 [843]</td>
</tr>
<tr>
<td></td>
<td>F_VN_ContainerAverageVarianceElementwise4 [848]</td>
</tr>
</tbody>
</table>

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64)</td>
<td>Tc3_Vision</td>
</tr>
<tr>
<td></td>
<td>with PL50, e.g.</td>
<td>Intel 4-core Atom CPU</td>
</tr>
</tbody>
</table>
6.4.7.8  F_VN_ContainerAverageVariance

**F_VN_ContainerAverageVariance**

- **ipSrcContainer**: ITCvNContainer
- **fAverage**: Reference To LREAL
- **fVariance**: Reference To LREAL
- **hrPrev**: HRESULT

Calculate the average and variance of elements in a container.

**Syntax**

Definition:

```c
FUNCTION F_VN_ContainerAverageVariance : HRESULT
VAR_INPUT
    ipSrcContainer : ITCvNContainer;
    fAverage : Reference To LREAL;
    fVariance : Reference To LREAL;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcContainer</td>
<td>ITCvNContainer [345]</td>
<td>Source container (Arithmetic, point and vector element types are supported. For points and vectors, the euclidean norm is used.)</td>
</tr>
<tr>
<td>fAverage</td>
<td>Reference To LREAL</td>
<td>Returns the calculated average value</td>
</tr>
<tr>
<td>fVariance</td>
<td>Reference To LREAL</td>
<td>Returns the calculated variance value</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Further information**

The table below shows which functions can be used for which container type:

<table>
<thead>
<tr>
<th>Types of containers</th>
<th>Usable functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Containers of elements</td>
<td>F_VN_ContainerAverage [840]</td>
</tr>
<tr>
<td></td>
<td>F_VN_ContainerAverageVariance [845]</td>
</tr>
<tr>
<td>Containers of vectors, each with 2 elements</td>
<td>F_VN_ContainerAverageElementwise2 [841]</td>
</tr>
<tr>
<td></td>
<td>F_VN_ContainerAverageVarianceElementwise2 [846]</td>
</tr>
<tr>
<td>Containers of vectors, each with 3 elements</td>
<td>F_VN_ContainerAverageElementwise3 [842]</td>
</tr>
<tr>
<td></td>
<td>F_VN_ContainerAverageVarianceElementwise3 [847]</td>
</tr>
<tr>
<td>Containers of vectors, each with 4 elements</td>
<td>F_VN_ContainerAverageElementwise4 [843]</td>
</tr>
<tr>
<td></td>
<td>F_VN_ContainerAverageVarianceElementwise4 [848]</td>
</tr>
</tbody>
</table>

**Required License**

TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.7.9 F_VN_ContainerAverageVarianceElementwise2

Calculate the elementwise average and variance of 2D elements in a container.

#### Syntax

**Definition:**

```c
FUNCTION F_VN_ContainerAverageVarianceElementwise2 : HRESULT
VAR_INPUT
   ipSrcContainer : ITcVnContainer;
   aAverage      : Reference To TcVnVector2_LREAL;
   aVariance     : Reference To TcVnVector2_LREAL;
   hrPrev        : HRESULT;
END_VAR
```

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcContainer</td>
<td>ITcVnContainer</td>
<td>Source container with 2D points or vectors</td>
</tr>
<tr>
<td>aAverage</td>
<td>Reference To TcVnVector2_LREAL</td>
<td>Returns the calculated average values</td>
</tr>
<tr>
<td>aVariance</td>
<td>Reference To TcVnVector2_LREAL</td>
<td>Returns the calculated variance values</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

#### Return value

**HRESULT [ ▶ 135]**

#### Further information

The table below shows which functions can be used for which container type:
Types of containers | Usable functions
--- | ---
Containers of elements | F_VN_ContainerAverage | F_VN_ContainerAverageVariance
Containers of vectors, each with 2 elements | F_VN_ContainerAverageElementwise2 | F_VN_ContainerAverageVarianceElementwise2
Containers of vectors, each with 3 elements | F_VN_ContainerAverageElementwise3 | F_VN_ContainerAverageVarianceElementwise3
Containers of vectors, each with 4 elements | F_VN_ContainerAverageElementwise4 | F_VN_ContainerAverageVarianceElementwise4

Required License
TC3 Vision Base

System Requirements
<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.7.10 F_VN_ContainerAverageVarianceElementwise3

### Syntax

**Definition:**

```
FUNCTION F_VN_ContainerAverageVarianceElementwise3 : HRESULT
VAR_INPUT
    ipSrcContainer : ITcVnContainer;
    aAverage : Reference To TcVnVector3_LREAL;
    aVariance : Reference To TcVnVector3_LREAL;
    hrPrev : HRESULT;
END_VAR
```

Calculate the elementwise average and variance of 3D elements in a container.

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container with 3D points or vectors</td>
</tr>
<tr>
<td>aAverage</td>
<td>Reference To TcVnVector3_LREAL [153]</td>
<td>Returns the calculated average values</td>
</tr>
<tr>
<td>aVariance</td>
<td>Reference To TcVnVector3_LREAL [153]</td>
<td>Returns the calculated variance values</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
**Return value**

HRESULT [135]

**Further information**

The table below shows which functions can be used for which container type:

<table>
<thead>
<tr>
<th>Types of containers</th>
<th>Usable functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Containers of elements</td>
<td>F_VN_ContainerAverage [840]</td>
</tr>
<tr>
<td></td>
<td>F_VN_ContainerAverageVariance [845]</td>
</tr>
<tr>
<td>Containers of vectors, each with 2 elements</td>
<td>F_VN_ContainerAverageElementwise2 [841]</td>
</tr>
<tr>
<td></td>
<td>F_VN_ContainerAverageVarianceElementwise2 [846]</td>
</tr>
<tr>
<td>Containers of vectors, each with 3 elements</td>
<td>F_VN_ContainerAverageElementwise3 [842]</td>
</tr>
<tr>
<td></td>
<td>F_VN_ContainerAverageVarianceElementwise3 [847]</td>
</tr>
<tr>
<td>Containers of vectors, each with 4 elements</td>
<td>F_VN_ContainerAverageElementwise4 [843]</td>
</tr>
<tr>
<td></td>
<td>F_VN_ContainerAverageVarianceElementwise4 [848]</td>
</tr>
</tbody>
</table>

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g.</td>
<td>Tc3_Vision</td>
</tr>
<tr>
<td></td>
<td>Intel 4-core Atom CPU</td>
<td></td>
</tr>
</tbody>
</table>

**6.4.7.11 F_VN_ContainerAverageVarianceElementwise4**

Calculate the elementwise average and variance of 4D elements in a container.

**Syntax**

**Definition:**

FUNCTION F_VN_ContainerAverageVarianceElementwise4 : HRESULT
VAR_INPUT
   ipSrcContainer : ITCvVContainer;
   aAverage       : Reference To TcVnVector4_LREAL;
   aVariance      : Reference To TcVnVector4_LREAL;
   hrPrev          : HRESULT;
END_VAR
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcContainer</td>
<td>ITcVnContainer</td>
<td>Source container with 4D vectors</td>
</tr>
<tr>
<td>aAverage</td>
<td>Reference To TcVnVector4 LREAL</td>
<td>Returns the calculated average values</td>
</tr>
<tr>
<td>aVariance</td>
<td>Reference To TcVnVector4 LREAL</td>
<td>Returns the calculated variance values</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Further information

The table below shows which functions can be used for which container type:

<table>
<thead>
<tr>
<th>Types of containers</th>
<th>Usable functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Containers of elements</td>
<td>F_VN_ContainerAverage [840]</td>
</tr>
<tr>
<td></td>
<td>F_VN_ContainerAverageVariance [845]</td>
</tr>
<tr>
<td>Containers of vectors, each with 2 elements</td>
<td>F_VN_ContainerAverageElementwise2 [841]</td>
</tr>
<tr>
<td></td>
<td>F_VN_ContainerAverageVarianceElementwise2 [846]</td>
</tr>
<tr>
<td>Containers of vectors, each with 3 elements</td>
<td>F_VN_ContainerAverageElementwise3 [842]</td>
</tr>
<tr>
<td></td>
<td>F_VN_ContainerAverageVarianceElementwise3 [847]</td>
</tr>
<tr>
<td>Containers of vectors, each with 4 elements</td>
<td>F_VN_ContainerAverageElementwise4 [843]</td>
</tr>
<tr>
<td></td>
<td>F_VN_ContainerAverageVarianceElementwise4 [848]</td>
</tr>
</tbody>
</table>

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.7.12 F_VN_MaxElement

Gets the maximum element of a container.
API reference

Syntax

Definition:

```
FUNCTION F_VN_MaxElement : HRESULT
VAR_INPUT
    ipSrcContainer : ITcVnContainer;
    ipDestIterator : Reference To ITcVnBidirectionalIterator;
    nIndex         : Reference To ULINT;
    hrPrev         : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container (Arithmetic, point and vector element types are supported. For points and vectors, the euclidean norm is used.)</td>
</tr>
<tr>
<td>ipDestIterator</td>
<td>Reference To ITcVnBidirectionalIterator [334]</td>
<td>Returns an iterator to the requested element</td>
</tr>
<tr>
<td>nIndex</td>
<td>Reference To ULINT</td>
<td>Returns the position of the requested element within the container</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.7.13  F_VN_MedianElement

```
F_VN_MedianElement
     ipSrcContainer  ITcVnContainer
     ipDestIterator  Reference To ITcVnBidirectionalIterator
     nIndex         Reference To ULINT
     hrPrev         HRESULT
```

Gets the median element of a container.

Syntax

Definition:

```
FUNCTION F_VN_MedianElement : HRESULT
VAR_INPUT
    ipSrcContainer : ITcVnContainer;
    ipDestIterator : Reference To ITcVnBidirectionalIterator;
    nIndex         : Reference To ULINT;
    hrPrev         : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcContainer</td>
<td>ITcVnContainer [345]</td>
<td>Source container (Arithmetic, point and vector element types are supported. For points and vectors, the euclidean norm is used.)</td>
</tr>
<tr>
<td>ipDestIterator</td>
<td>Reference To ITcVnBidirectionalIterator [334]</td>
<td>Returns an iterator to the requested element</td>
</tr>
<tr>
<td>nIndex</td>
<td>Reference To UINT</td>
<td>Returns the position of the requested element within the container</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.7.14 F_VN_MinElement

```plaintext
FUNCTION F_VN_MinElement : HRESULT
VAR_INPUT
    ipSrcContainer : ITcVnContainer;
    ipDestIterator : Reference To ITcVnBidirectionalIterator;
    nIndex         : Reference To UINT;
    hrPrev          : HRESULT;
END_VAR
```

Gets the minimum element of a container.

## Syntax

**Definition:**

FUNCTION F_VN_MinElement : HRESULT
VAR_INPUT
    ipSrcContainer : ITcVnContainer;
    ipDestIterator : Reference To ITcVnBidirectionalIterator;
    nIndex         : Reference To UINT;
    hrPrev          : HRESULT;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcContainer</td>
<td>ITcVnContainer [852]</td>
<td>Source container (Arithmetic, point and vector element types are supported. For points and vectors, the euclidean norm is used.)</td>
</tr>
<tr>
<td>ipDestIterator</td>
<td>Reference To ITcVnBidirectionalIterator [334]</td>
<td>Returns an iterator to the requested element</td>
</tr>
<tr>
<td>nIndex</td>
<td>Reference To ULINT</td>
<td>Returns the position of the requested element within the container</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.8 Contour Analysis

The group contains functions for the analysis of contours [146]:

### Functions

#### Shape approximation

- F_VN_ApproximatePolygon [853]
- F_VN_ConvexHullPoints(Exp) [863]
- F_VN_EnclosingCircle [866]
- F_VN_EnclosingRectangle [867]
- F_VN_EnclosingTriangle [868]
- F_VN_FitEllipse [869]
- F_VN_FitLine(Exp) [870]
- F_VN_UprightBoundingRectangle [877]

#### Points

- F_VN_CheckIfPointIsInsideContour [854]
- F_VN_ContourExtremePoint [858]
- F_VN_ConvexityDefects [864]

#### Geometrical features

- F_VN_ContourArea [855]
F_VN_ApproximatePolygon

F_VN_ApproximatePolygon

```
FUNCTION F_VN_ApproximatePolygon : HRESULT
VAR_INPUT
  ipSrcContour : ITcVnContainer;
  ipDestContour : Reference To ITcVnContainer;
  fMaxDist     : LREAL;
  bClosed      : BOOL;
  hrPrev       : HRESULT;
END_VAR
```

Approximate a contour to a simplified polygon (using the Douglas-Peucker algorithm).

**Syntax**

**Definition:**

FUNCTION F_VN_ApproximatePolygon : HRESULT

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcContour</td>
<td>ITcVnContainer [345]</td>
<td>Single contour (ContainerType_Vector_TcVnPoint2_DINT or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ContainerType_Vector_TcVnPoint2_REAL; The elements of this</td>
</tr>
<tr>
<td></td>
<td></td>
<td>container are the points of the contour.)</td>
</tr>
<tr>
<td>ipDestContour</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container which is filled with the approximated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>polygon points (same type ID as ipSrcContour; Non-zero</td>
</tr>
<tr>
<td></td>
<td></td>
<td>interface pointers are reused.)</td>
</tr>
<tr>
<td>fMaxDist</td>
<td>LREAL</td>
<td>Maximum distance between the original contour and its</td>
</tr>
<tr>
<td>bClosed</td>
<td>BOOL</td>
<td>Specify, if the contour is closed (first and last points connected) or</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.8.2 F_VN_CheckIfPointIsInsideContour

Checks if a point is inside a contour (and optionally return the distance to it).

Syntax

Definition:

FUNCTION F_VN_CheckIfPointIsInsideContour : HRESULT
VAR_INPUT
    ipContour : ITCvnContainer;
    aPoint : Reference To TcVnPoint2_REAL;
    fDist : Reference To LREAL;
    bMeasureDistance : BOOL;
    hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContour</td>
<td>ITCvnContainer [345]</td>
<td>Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>aPoint</td>
<td>Reference To TcVnPoint2_REAL [151]</td>
<td>Point position to check</td>
</tr>
<tr>
<td>fDist</td>
<td>Reference To LREAL</td>
<td>Returns the result (&gt; 0: the point is inside the contour; 0: the point is on the contour; &lt; 0: the point is outside the contour)</td>
</tr>
<tr>
<td>bMeasureDistance</td>
<td>BOOL</td>
<td>If true, fDist returns the distance to the nearest contour edge. Otherwise, fDist only returns -1, 0 or +1 (recommended to set to false if the distance is not required, as this is faster).</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]
### F_VN_ContourArea

Estimate the contour area using Green's formula.

#### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_ContourArea : HRESULT
VAR_INPUT
    ipContour : ITCvNContainer;
    fArea      : Reference To LREAL;
    hrPrev     : HRESULT;
END_VAR
```

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContour</td>
<td>ITCvNContainer [345]</td>
<td>Single contour (ContainerType_Vector_TcVNPoint2_DINT or ContainerType_Vector_TcVNPoint2_REAL; The elements of this container are the points of the contour.)</td>
</tr>
<tr>
<td>fArea</td>
<td>Reference To LREAL</td>
<td>Returns the estimated contour area (The actual contour area may differ depending on the shape of the contour.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

#### Return value

**HRESULT [135]**

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.8.4 F_VN_ContourCenterOfMass

Computes the center of mass of a contour.

**Syntax**

**Definition:**

\[
\text{FUNCTION F_VN_ContourCenterOfMass : HRESULT} \\
\text{VAR_INPUT} \\
\hspace{1em} \text{ipContour : ITCvNContainer;} \\
\hspace{1em} \text{aCenterOfMass : Reference To TcVnPoint2_LREAL;} \\
\hspace{1em} \text{hrPrev : HRESULT;} \\
\text{END_VAR} \\
\]

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContour</td>
<td>ITCvNContainer</td>
<td>Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)</td>
</tr>
<tr>
<td>aCenterOfMass</td>
<td>Reference To TcVnPoint2_LREAL</td>
<td>Returns the center of mass of the contour</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.8.5 F_VN_ContourCircularity

Computes the circularity of a contour (1.0: ideal circle, 0.0: straight line).

**Syntax**

**Definition:**

\[
\text{FUNCTION F_VN_ContourCircularity : HRESULT} \\
\text{VAR_INPUT} \\
\hspace{1em} \text{ipContour : ITCvNContainer;} \\
\hspace{1em} \text{fCircularity : Reference To LREAL;} \\
\hspace{1em} \text{hrPrev : HRESULT;} \\
\text{END_VAR} \\
\]
FUNCTION F_VN_ContourCircularity : HRESULT
VAR_INPUT
    ipContour : ITcVnContainer;
    fCircularity : Reference To LREAL;
    hrPrev : HRESULT;
END_VAR

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContour</td>
<td>ITcVnContainer [345]</td>
<td>Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)</td>
</tr>
<tr>
<td>fCircularity</td>
<td>Reference To LREAL</td>
<td>Returns the circularity of the contour [0..1]</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.8.6 F_VN_ContourEccentricity

Computes the eccentricity of a contour (0.0: circular, 1.0: linear).

### Syntax

**Definition:**

FUNCTION F_VN_ContourEccentricity : HRESULT
VAR_INPUT
    ipContour : ITcVnContainer;
    fEccentricity : Reference To LREAL;
    hrPrev : HRESULT;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContour</td>
<td>ITCvNContainer [345]</td>
<td>Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)</td>
</tr>
<tr>
<td>fEccentricity</td>
<td>Reference To LREAL</td>
<td>Returns the eccentricity of the contour [0..1]</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.8.7 F_VN_ContourExtremePoint

Finds the contour extreme point.

### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_ContourExtremePoint : HRESULT
VAR_INPUT
    ipContour : ITCvNContainer;
    eDirection : ETcVnExtremePointDirection;
    aExtremePoint : Reference To TcVnPoint2_LREAL;
    hrPrev : HRESULT;
END_VAR
```

---

**F_VN_ContourExtremePoint**

- **ipContour ITCvNContainer**
- **eDirection ETcVnExtremePointDirection**
- **aExtremePoint Reference To TcVnPoint2_LREAL**
- **hrPrev HRESULT**

Finds the contour extreme point.
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContour</td>
<td>ITcVnContainer [345]</td>
<td>Source container with 2D points (TcVnPoint2_DINT or TcVnPoint2_REAL or TcVnPoint2_LREAL)</td>
</tr>
<tr>
<td>eDirection</td>
<td>ETcVnExtremePointDirection [187]</td>
<td>Selects the search direction for the extreme point</td>
</tr>
<tr>
<td>aExtremePoint</td>
<td>Reference To TcVnPoint2_LREAL [151]</td>
<td>Returns the extreme point of the contour, according to eDirection</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.8.8 F_VN_ContourMoments

**F_VN_ContourMoments**

- **ipContour**: ITcVnContainer
- **stMoments**: Reference To TcVnMoments
- **hrPrev**: HRESULT

Computes the spatial moments, the central moments, and the central normalized moments of a contour up to the third order.

### Syntax

**Definition:**

```c
FUNCTION F_VN_ContourMoments : HRESULT
VAR_INPUT
   ipContour : ITcVnContainer;
   stMoments : Reference To TcVnMoments;
   hrPrev : HRESULT;
END_VAR
```
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContour</td>
<td>ITcVnContainer [345]</td>
<td>Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)</td>
</tr>
<tr>
<td>stMoments</td>
<td>Reference To TcVnMoments [209]</td>
<td>Returns a struct containing the moments</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.8.9 F_VN_ContourPerimeter

F_VN_ContourPerimeter

- **ipContour**: ITcVnContainer
- **fPerimeter**: Reference To LREAL
- **bClosed**: BOOL
- **hrPrev**: HRESULT

Computes the perimeter of a contour (curve length if the contour is not closed).

### Syntax

**Definition:**

FUNCTION F_VN_ContourPerimeter : HRESULT
VAR_INPUT
  ipContour : ITcVnContainer;
  fPerimeter : Reference To LREAL;
  bClosed : BOOL;
  hrPrev : HRESULT;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ipContour</strong></td>
<td>ITCvNContainer [345]</td>
<td>Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)</td>
</tr>
<tr>
<td><strong>fPerimeter</strong></td>
<td>Reference To LREAL</td>
<td>Returns the perimeter of the contour</td>
</tr>
<tr>
<td><strong>bClosed</strong></td>
<td>BOOL</td>
<td>Specifies, if the contour is closed (first and last points connected) or not</td>
</tr>
<tr>
<td><strong>hrPrev</strong></td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

**HRESULT [135]**

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.8.10  F_VN_ContourRoundness

**F_VN_ContourRoundness**

```plaintext
ipContour  ITCvNContainer
fRoundness Reference To LREAL
hrPrev     HRESULT
```

Computes the roundness of a contour \((\text{perimeter}^2 / \text{area})\).

**Syntax**

Definition:

```plaintext
FUNCTION F_VN_ContourRoundness : HRESULT
VAR_INPUT
  ipContour  : ITCvNContainer;
  fRoundness : Reference To LREAL;
  hrPrev     : HRESULT;
END_VAR
```

## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ipContour</strong></td>
<td>ITCvNContainer [345]</td>
<td>Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)</td>
</tr>
<tr>
<td><strong>fRoundness</strong></td>
<td>Reference To LREAL</td>
<td>Returns the roundness of the contour</td>
</tr>
<tr>
<td><strong>hrPrev</strong></td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.8.11 F_VN_ConvexHullPoints

```
FUNCTION F_VN_ConvexHullPoints : HRESULT
VAR_INPUT
    ipPointSet : ITcVnContainer;
    ipConvexHull : Reference To ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Determines the convex hull of a point set.

Syntax

Definition:

```
FUNCTION F_VN_ConvexHullPoints : HRESULT
VAR_INPUT
    ipPointSet : ITcVnContainer;
    ipConvexHull : Reference To ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipPointSet</td>
<td>ITcVnContainer [345]</td>
<td>Source point set (ContainerType_Vector_TcVnPoint2_DINT or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ContainerType_Vector_TcVnPoint2_REAL; The elements of this</td>
</tr>
<tr>
<td></td>
<td></td>
<td>container are the individual points.)</td>
</tr>
<tr>
<td>ipConvexHull</td>
<td>Reference To ITcVnContainer</td>
<td>Returns a container which is filled with the sorted convex hull</td>
</tr>
<tr>
<td></td>
<td></td>
<td>points (same type ID as ipPointSet; Non-zero interface pointers</td>
</tr>
<tr>
<td></td>
<td></td>
<td>are reused.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.8.12  F_VN_ConvexHullPointsExp

**F_VN_ConvexHullPointsExp**

```
ipPointSet  ITcVnContainer
ipConvexHull Reference To ITcVnContainer
bClockwise  BOOL
hrPrev      HRESULT
```

Determines the convex hull of a point set. (expert function)

**Syntax**

**Definition:**

```
FUNCTION F_VN_ConvexHullPointsExp : HRESULT
VAR_INPUT
  ipPointSet   : ITcVnContainer;
  ipConvexHull : Reference To ITcVnContainer;
  bClockwise  : BOOL;
  hrPrev      : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipPointSet</td>
<td>ITcVnContainer [345]</td>
<td>Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)</td>
</tr>
<tr>
<td>ipConvexHull</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container which is filled with the sorted convex hull points (same type ID as ipPointSet; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>bClockwise</td>
<td>BOOL</td>
<td>Selects the sorting direction of the hull points, assuming cartesian coordinates.</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

TF7000 - TF7300  Version: 1.3  863
6.4.8.13  F_VN_ConvexityDefects

F_VN_ConvexityDefects

- ipContour  ITcVnContainer
- ipConvexityDefects  Reference To ITcVnContainer
- hrPrev  HRESULT

Finds the convexity defects of a point set.

Syntax

Definition:

FUNCTION F_VN_ConvexityDefects : HRESULT
VAR_INPUT
  ipContour  : ITcVnContainer;
  ipConvexityDefects  : Reference To ITcVnContainer;
  hrPrev  : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContour</td>
<td>ITcVnContainer</td>
<td>Source contour (ContainerType_Vector_TcVnPoint2_DINT; The elements of this container are the individual points.)</td>
</tr>
<tr>
<td>ipConvexityDefects</td>
<td>Reference To ITcVnContainer</td>
<td>Returns a container which is filled with the convexity defects (ContainerType_Vector_TcVnVector4_DINT with [startIndex, endIndex, furthestPointIndex, fixedPointDistance (8 fraction bits, i.e. divide by 256 to get the distance in pixels)]; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.8.14  F_VN_CustomElementWiseContainerOperation_ITcVnContainer

F_VN_CustomElementWiseContainerOperation_ITcVnContainer

- ipContainer1  ITcVnContainer
- ipContainer2  ITcVnContainer
- ipContainer3  ITcVnContainer
- ipOperationFB  ITcVnCustomContainerOperation_ITcVnContainer
- hrPrev  HRESULT

Performs a custom, element wise operation on a set of up to 3 containers.

Syntax

Definition:
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer1</td>
<td>ITcVnContainer [345]</td>
<td>Container 1</td>
</tr>
<tr>
<td>ipContainer2</td>
<td>ITcVnContainer [345]</td>
<td>Container 2</td>
</tr>
<tr>
<td>ipContainer3</td>
<td>ITcVnContainer [345]</td>
<td>Container 3</td>
</tr>
<tr>
<td>ipOperationFB</td>
<td>ITcVnCustomContainerOperation_ITcVnContainer [224]</td>
<td>Custom operation on the container elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.8.15 F_VN_CustomElementWiseContainerOperation_ITcVnForwardIterator**

Performs a custom, element wise operation on a set of up to 3 containers.

**Syntax**

Definition:

FUNCTION F_VN_CustomElementWiseContainerOperation_ITcVnForwardIterator : HRESULT

VAR_INPUT

  ipContainer1 : ITcVnContainer;
  ipContainer2 : ITcVnContainer;
  ipContainer3 : ITcVnContainer;
  ipOperationFB : ITcVnCustomContainerOperation_ITcVnForwardIterator;
  hrPrev : HRESULT;
END_VAR
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer1</td>
<td>ITcVnContainer [345]</td>
<td>Container 1</td>
</tr>
<tr>
<td>ipContainer2</td>
<td>ITcVnContainer [345]</td>
<td>Container 2</td>
</tr>
<tr>
<td>ipContainer3</td>
<td>ITcVnContainer [345]</td>
<td>Container 3</td>
</tr>
<tr>
<td>ipOperationFB</td>
<td>ITcVnCustomContainerOp</td>
<td>Custom operation on the elements</td>
</tr>
<tr>
<td></td>
<td>operation ITcVnForwardItera-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>tor [225]</td>
<td></td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.8.16 **F_VN_EnclosingCircle**

Searchs for a minimum area circle enclosing a set of points.

**Syntax**

**Definition:**

FUNCTION F_VN_EnclosingCircle : HRESULT
VAR_INPUT
  ipPointSet : ITcVnContainer;
  aCenter : Reference To TcVnPoint2_REAL;
  fRadius : Reference To REAL;
  hrPrev : HRESULT;
END_VAR
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipPointSet</td>
<td>ITcVnContainer ![345]</td>
<td>Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)</td>
</tr>
<tr>
<td>aCenter</td>
<td>Reference To TcVnPoint2_REAL ![151]</td>
<td>Returns the center of the circle</td>
</tr>
<tr>
<td>fRadius</td>
<td>Reference To REAL</td>
<td>Returns the radius of the circle</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT ![135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT ![135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.8.17 F_VN_EnclosingRectangle**

Searches for a minimum area rectangle enclosing a set of points.

**Syntax**

**Definition:**

FUNCTION F_VN_EnclosingRectangle : HRESULT
VAR_INPUT
    ipPointSet : ITcVnContainer;
    stRectangle : Reference To TcVnRotatedRectangle;
    hrPrev : HRESULT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipPointSet</td>
<td>ITcVnContainer ![345]</td>
<td>Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)</td>
</tr>
<tr>
<td>stRectangle</td>
<td>Reference To TcVnRotatedRectangle ![222]</td>
<td>Returns the determined rectangle</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT ![135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.8.18 F_VN_EnclosingTriangle

Searches for a minimum area triangle enclosing a set of points.

Syntax

Definition:

FUNCTION F_VN_EnclosingTriangle : HRESULT
VAR_INPUT
  ipPointF : ITCvNContainer;
  aTriangleVertices : Reference To TcVnArray3_Point2_REAL;
  hrPrev : HRESULT;
END_VAR
VAR_OUTPUT
  fArea : LREAL;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipPointF</td>
<td>ITCvNContainer [345]</td>
<td>Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)</td>
</tr>
<tr>
<td>aTriangleVertices</td>
<td>Reference To TcVnArray3_Point2_REAL [153]</td>
<td>Returns the 3 triangle vertices</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fArea</td>
<td>LREAL</td>
<td>Returns the triangle area</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]
6.4.8.19  F_VN_FitEllipse

Fit an ellipse in a point set.

Syntax

Definition:

FUNCTION F_VN_FitEllipse : HRESULT
VAR_INPUT
  ipPointSet : ITCvNContainer;
  stEllipse : Reference To TcVnRotatedRectangle;
  hrPrev   : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipPointSet</td>
<td>ITCvNContainer [345]</td>
<td>Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; At least 5 reasonable points are required! The elements of this container are the individual points.)</td>
</tr>
<tr>
<td>stEllipse</td>
<td>Reference To TcVnRotatedRectangle [222]</td>
<td>Resulting ellipse, described by a rotated rectangle.</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.8.20  F_VN_FitLine

**F_VN_FitLine**

- ipPointSet : ITcVnContainer
- aFitLine : Reference To TcVnVector4_LREAL
- hrPrev : HRESULT

Fit a line into a point set.

**Syntax**

Definition:

```c
FUNCTION F_VN_FitLine : HRESULT
VAR_INPUT
    ipPointSet : ITcVnContainer;
    aFitLine : Reference To TcVnVector4_LREAL;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipPointSet</td>
<td>ITcVnContainer [345]</td>
<td>Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)</td>
</tr>
<tr>
<td>aFitLine</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Resulting line. The first and second element describe the x and y component of a vector collinear to the line. The third and fourth element describe the x and y component of a point on the line.</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.8.21  F_VN_FitLineExp

**F_VN_FitLineExp**

- ipPointSet : ITcVnContainer
- aFitLine : Reference To TcVnVector4_LREAL
- eDistanceType : ETCvnDistanceType
- fParam : LREAL
- fEpsRadius : LREAL
- fEpsAngle : LREAL
- hrPrev : HRESULT

This function is an extension of F_VN_FitLine, allowing for additional parameters to control the fitting process.
Fit a line into a point set. (expert function)

**Syntax**

**Definition:**

FUNCTION F_VN_FitLineExp : HRESULT
VAR_INPUT
    ipPointSet : ITcVnContainer;
    aFitLine   : Reference To TcVnVector4_LREAL;
    eDistanceType : ETcVnDistanceType;
    fParam     : LREAL;
    fEpsRadius : LREAL;
    fEpsAngle  : LREAL;
    hrPrev     : HRESULT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipPointSet</td>
<td>ITcVnContainer</td>
<td>Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)</td>
</tr>
<tr>
<td>aFitLine</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>Resulting line. The first and second element describe the x and y component of a vector collinear to the line. The third and fourth element describe the x and y component of a point on the line.</td>
</tr>
<tr>
<td>eDistanceType</td>
<td>ETcVnDistanceType</td>
<td>Distance computation method (supported: L1, L2, L12, FAIR, WELSCH, HUBER)</td>
</tr>
<tr>
<td>fParam</td>
<td>LREAL</td>
<td>Numerical parameter (c) for some eDistanceType (should be &gt;= 0). If 0, an optimal value is chosen internally.</td>
</tr>
<tr>
<td>fEpsRadius</td>
<td>LREAL</td>
<td>Accuracy of the radius (distance of the line from the coordinate origin, should be &gt; 0). A smaller value means higher accuracy.</td>
</tr>
<tr>
<td>fEpsAngle</td>
<td>LREAL</td>
<td>Accuracy of the angle (should be &gt; 0). A smaller value means higher accuracy.</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.8.22 **F_VN_FourierDescriptors**

**F_VN_FourierDescriptors**

```
FUNCTION F_VN_FourierDescriptors : HRESULT
VAR_INPUT
    ipContour : ITcVnContainer;
    ipDescriptors : Reference To ITcVnContainer;
    nDescriptors : DINT;
    hrPrev : HRESULT;
END_VAR
```
Computes the fourier descriptors for a closed contour.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_FourierDescriptors : HRESULT
VAR_INPUT
    ipContour     : ITcVnContainer;
    ipDescriptors : Reference To ITcVnContainer;
    nDescriptors  : DINT;
    hrPrev        : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContour</td>
<td>ITcVnContainer [345]</td>
<td>Closed contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL. Providing the full contour is strongly recommended, i.e. use TCVN_CAM_NONE for contour detection algorithms.)</td>
</tr>
<tr>
<td>ipDescriptors</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns the fourier descriptors for ipContour (Re0, Im0, Re1, Im1, ...; ContainerType_Vector_LREAL if ipContour is of type ContainerType_Vector_TcVnPoint2_LREAL, else ContainerType_Vector_REAL.)</td>
</tr>
<tr>
<td>nDescriptors</td>
<td>DINT</td>
<td>Specifies how many descriptors should be returned (set to -1 to return all computed descriptors, i.e. at least as many as the number of contour points)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.8.23 F_VN_MatchContours**

```c
F_VN_MatchContours
IpContour1  ITcVnContainer
IpContour2  ITcVnContainer
EComparisonMethod  ETcVnContoursMatchComparisonMethod
FDissimilarity  Reference To LREAL
HrPrev      HRESULT
```

Compare contours using the Hu moment invariants. In case of multiple contours in each container, the best matches are found and the average dissimilarity over all matched contours is returned.
Syntax

Definition:

FUNCTION F_VN_MatchContours : HRESULT
VAR_INPUT
  ipContour1 : ITcVnContainer;
  ipContour2 : ITcVnContainer;
  eComparisonMethod : ETcVnContoursMatchComparisonMethod;
  fDissimilarity : Reference To LREAL;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContour1</td>
<td>ITcVnContainer</td>
<td>First contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour) or collection of multiple contours (ContainerType_Vector_Vector_TcVnPoint2_DINT)</td>
</tr>
<tr>
<td>ipContour2</td>
<td>ITcVnContainer</td>
<td>Second contour (same type as ipContour1)</td>
</tr>
<tr>
<td>eComparisonMethod</td>
<td>ETcVnContoursMatchComparisonMethod</td>
<td>Method used for comparing the Hu moment invariants of the contours</td>
</tr>
<tr>
<td>fDissimilarity</td>
<td>Reference To LREAL</td>
<td>Returns the dissimilarity of the contours depending on the chosen comparison method</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [ 135 ]

Required License

TC3 Vision Matching

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.8.24  F_VN_MatchContours1vsN

FUNCTION F_VN_MatchContours1vsN : HRESULT
VAR_INPUT
  ipRefContour : ITcVnContainer;
  ipContours : ITcVnContainer;
  ipMatchIndexes : Reference To ITcVnContainer;
  ipDissimilarities : Reference To ITcVnContainer;
  fDissimilarityThreshold : LREAL;
  eComparisonMethod : ETcVnContoursMatchComparisonMethod;
  hrPrev : HRESULT;
END_VAR

Compare a reference contour with multiple other contours using the Hu moment invariants. Returns a sorted list of best matches.
Syntax

Definition:

FUNCTION F_VN_MatchContours1vsN : HRESULT
VAR_INPUT
  ipRefContour : ITcVnContainer;
  ipContours : ITcVnContainer;
  ipMatchIndexes : Reference To ITcVnContainer;
  ipDissimilarities : Reference To ITcVnContainer;
  fDissimilarityThreshold : LREAL;
  eComparisonMethod : ETcVnContoursMatchComparisonMethod;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| ipRefContour    | ITcVnContainer                      | Reference contour (ContainerType_Vector_TcVnPoint2_DINT or
|                 |                                     | ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the
|                 |                                     | points of the contour)                                                      |
| ipContours      | ITcVnContainer                      | Collection of multiple contours (CTcVnContainer_Vector_Vector_TcVnPoint2_DINT) |
| ipMatchIndexes  | Reference To ITcVnContainer          | Returns the indexes of the best matches (CTcVnContainer_Vector_ULINT; sorted,
|                 |                                     | first element is best match, i.e. has lowest dissimilarity)                 |
| ipDissimilarities | Reference To ITcVnContainer         | Returns the computed dissimilarities of the best matches
|                  |                                     | (CTcVnContainer_Vector_LREAL; Optional, set to 0 if not required; sorted
|                  |                                     | corresponding to ipMatchIndexes)                                           |
| fDissimilarityThreshold | LREAL                           | Neglect irrelevant matches, i.e. dissimilarity > fDissimilarityThreshold |
| eComparisonMethod | ETcVnContoursMatchComparisonMethod | Method used for comparing the Hu moment invariants of the contours       |
| hrPrev          | HRESULT                             | HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev)
|                 |                                     | equals false, no operation is executed.)                                  |

Return value

HRESULT [135]

Required License

TC3 Vision Matching

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
F_VN_MatchContours1vsNExp

**F_VN_MatchContours1vsNExp**

- **ipRefContour**: ITCvntContainer
- **ipContours**: ITCvntContainer
- **ipMatchIndexes**: Reference To ITCvntContainer
- **ipDissimilarities**: Reference To ITCvntContainer
- **fDissimilarityThreshold**: LREAL
- **eComparisonMethod**: ETCvntContoursMatchComparisonMethod
- **fAreaFactor**: LREAL
- **fAbsPositionFactor**: LREAL
- **hrPrev**: HRESULT

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipRefContour</td>
<td>ITCvntContainer [345]</td>
<td>Reference contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour)</td>
</tr>
<tr>
<td>ipContours</td>
<td>ITCvntContainer [345]</td>
<td>Collection of multiple contours (CTcVntContainer_Vector_Vector_TcVnPoint2_DINT)</td>
</tr>
<tr>
<td>ipMatchIndexes</td>
<td>Reference To ITCvntContainer [345]</td>
<td>Returns the indexes of the best matches (CTcVntContainer_Vector_ULINT; sorted, first element is best match, i.e. has lowest dissimilarity)</td>
</tr>
<tr>
<td>ipDissimilarities</td>
<td>Reference To ITCvntContainer [345]</td>
<td>Returns the computed dissimilarities of the best matches (CTcVntContainer_Vector_LREAL; Optional, set to 0 if not required; sorted corresponding to ipMatchIndexes)</td>
</tr>
<tr>
<td>fDissimilarityThreshold</td>
<td>LREAL</td>
<td>Neglect irrelevant matches, i.e. dissimilarity &gt; fDissimilarityThreshold</td>
</tr>
<tr>
<td>eComparisonMethod</td>
<td>ETCvntContoursMatchComparisonMethod [180]</td>
<td>Method used for comparing the Hu moment invariants of the contours</td>
</tr>
<tr>
<td>fAreaFactor</td>
<td>LREAL</td>
<td>If &gt; 0, the relative area difference between contours is scaled by this factor and added to the computed dissimilarity</td>
</tr>
<tr>
<td>fAbsPositionFactor</td>
<td>LREAL</td>
<td>If &gt; 0, the absolute position difference between contours (i.e. the coordinates of the geometric contour centers) is scaled by this factor and added to the computed dissimilarity</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_MatchContours1vsNExp : HRESULT
VAR_INPUT
  ipRefContour      : ITCvntContainer;
  ipContours        : ITCvntContainer;
  ipMatchIndexes    : Reference To ITCvntContainer;
  ipDissimilarities : Reference To ITCvntContainer;
  fDissimilarityThreshold : LREAL;
  eComparisonMethod : ETCvntContoursMatchComparisonMethod;
  fAreaFactor       : LREAL;
  fAbsPositionFactor: LREAL;
  hrPrev            : HRESULT;
END_VAR
```

Compare a reference contour with multiple other contours using the Hu moment invariants (and optionally further aspects). Returns a sorted list of best matches. (expert function)
Compare contours using the Hu moment invariants (and optionally further aspects). In case of multiple contours in each container, the best matches are found and the average dissimilarity over all matched contours is returned. (expert function)

**Syntax**

**Definition:**

```c
FUNCTION F_VN_MatchContoursExp : HRESULT
VAR_INPUT
    ipContour1 : ITcVnContainer;
    ipContour2 : ITcVnContainer;
    eComparisonMethod : ETCvNContoursMatchComparisonMethod;
    fDissimilarity : Reference To LREAL;
    fAreaFactor : LREAL;
    fAbsPositionFactor : LREAL;
    fRelPositionFactor : LREAL;
    fNumDiffFactor : LREAL;
    hrPrev : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContour1</td>
<td>ITCvNContainer</td>
<td>First contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour) or collection of multiple contours (ContainerType_Vector_Vector_TcVnPoint2_DINT)</td>
</tr>
<tr>
<td>ipContour2</td>
<td>ITCvNContainer</td>
<td>Second contour (same type as ipContour1)</td>
</tr>
<tr>
<td>Method</td>
<td>ETcVnContoursMatchComparisonMethod</td>
<td>Method used for comparing the Hu moment invariants of the contours</td>
</tr>
<tr>
<td>fdissimilarity</td>
<td>Reference To LREAL</td>
<td>Returns the dissimilarity of the contours depending on the chosen comparison method</td>
</tr>
<tr>
<td>fAreaFactor</td>
<td>LREAL</td>
<td>If &gt; 0, the relative area difference between contours is scaled by this factor and added to the computed dissimilarity</td>
</tr>
<tr>
<td>fAbsPositionFactor</td>
<td>LREAL</td>
<td>If &gt; 0, the absolute position difference between contours (i.e. the coordinates of the geometric contour centers) is scaled by this factor and added to the computed dissimilarity</td>
</tr>
<tr>
<td>fRelPositionFactor</td>
<td>LREAL</td>
<td>If &gt; 0, the relative position difference between contours is scaled by this factor and added to the computed dissimilarity (only used if matching multiple contours)</td>
</tr>
<tr>
<td>fNumDiffFactor</td>
<td>LREAL</td>
<td>If &gt; 0, the difference between the number of contours in both containers is scaled by this factor and added to the computed dissimilarity (only used if matching multiple contours)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Matching

## System Requirements

### Development environment

<table>
<thead>
<tr>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

# 6.4.8.27 F_VN_UprightBoundingRectangle

Determines the upright bounding rectangle of a set of points.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_UprightBoundingRectangle : HRESULT
VAR_INPUT
    ipPointSet : ITCvNContainer;
```
stRectangle : Reference To TcVnRectangle_UDINT;
hrPrev : HRESULT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipPointSet</td>
<td>ITCvNContainer [345]</td>
<td>Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)</td>
</tr>
<tr>
<td>stRectangle</td>
<td>Reference To TcVnRectangle_UDINT [222]</td>
<td>Returns the determined rectangle</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.9 Control Functions**

This group contains functions for sequential control, e.g. for the handling of watchdogs [137].

**Functions**

**Watchdogs**

- F_VN_StartAbsWatchdog(Exp) [879]
- F_VN_StartRelWatchdog(Exp) [881]
- F_VN_StopWatchdog [882]

**6.4.9.1 F_VN_StartAbsWatchdog**

```
F_VN_StartAbsWatchdog

hrPrev : HRESULT

hrPrev = HRESULT
```

Starts a cooperative watchdog given an absolute stop time.

**Syntax**

**Definition:**
FUNCTION F_VN_StartAbsWatchdog : HRESULT
VAR_INPUT
  tStop : DINT;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStop</td>
<td>DINT</td>
<td>Stop time in us</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT \[135\]

Further information

Further information can be found in the chapter Watchdogs \[137\].

- **tStop time exceeded**
  
  Note that the tStop time is slightly exceeded (by a few µs) when the watchdog is triggered.

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.9.2 F_VN_StartAbsWatchdogExp

Start a cooperative watchdog given an absolute stop time. (expert function)

Syntax

Definition:

FUNCTION F_VN_StartAbsWatchdogExp : HRESULT
VAR_INPUT
  tStop : DINT;
  eWatchdogAccType : ETcWatchdogAccumulationType;
  hrPrev : HRESULT;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStop</td>
<td>DINT</td>
<td>Stop time in us</td>
</tr>
<tr>
<td>eWatchdogAcc</td>
<td>ETcWatchdogAccumulationType [202]</td>
<td>Accumulation method used for combining the results of multiple functions enclosed by this watchdog</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

### Development environment
- TwinCAT V3.1.4024.17 or later

### Target platform
- PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU

### PLC libraries to include
- Tc3_Vision

## 6.4.9.3 F_VN_StartRelWatchdog

**F_VN_StartRelWatchdog**

- tStop : DINT
- hrPrev : HRESULT

Starts a cooperative watchdog given a stop time relative to the current time.

### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_StartRelWatchdog : HRESULT
VAR_INPUT
  tStop : DINT;
  hrPrev : HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStop</td>
<td>DINT</td>
<td>Stop time in us</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]
Further information

Further information can be found in the chapter Watchdogs [137].

- **tStop time exceeded**
  
  Note that the tStop time is slightly exceeded (by a few µs) when the watchdog is triggered.

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.9.4 F_VN_StartRelWatchdogExp

Start a cooperative watchdog given a stop time relative to the current time. (expert function)

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_StartRelWatchdogExp : HRESULT
VAR_INPUT
  tStop : DINT;
  eWatchdogAccType : ETcWatchdogAccumulationType;
  hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStop</td>
<td>DINT</td>
<td>Stop time in us</td>
</tr>
<tr>
<td>eWatchdogAccType</td>
<td>ETcWatchdogAccumulationType</td>
<td>Accumulation method used for combining the results of multiple functions enclosed by this watchdog</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

Required License

TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.9.5 F_VN_StopWatchdog

Stops a watchdog and provide runtime information.

Syntax

Definition:

```
FUNCTION F_VN_StopWatchdog : HRESULT
VAR_INPUT
    hrStartWatchdog : HRESULT;
END_VAR
VAR_OUTPUT
    nFunctionsMonitored : ULINT;
    nFractionProcessed : UDINT;
    tRest             : DINT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hrStartWatchdog</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of the function used to start the watchdog. (If SUCCEEDED(hrStartWatchdog) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nFunctionsMonitored</td>
<td>ULINT</td>
<td>Returns the number of functions monitored</td>
</tr>
<tr>
<td>nFractionProcessed</td>
<td>UDINT</td>
<td>Returns the fraction processed accumulated over the monitored functions in percent</td>
</tr>
<tr>
<td>tRest</td>
<td>DINT</td>
<td>Returns the remaining computation time in us (may be negative)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Further information**

The output nFractionProcessed must be interpreted differently depending on the parameterization of the watchdog. This is set with ETcWatchdogAccumulationType [202] when starting the watchdog.

Further information can be found under Watchdogs [137].
**6.4.10 Drawing**

This group contains functions for painting and describing images.

### Handling 4-channel colors

4-channel colors are accepted directly into the result image. No alpha blending or similar takes place.

### Functions

#### Circles and Arcs

- F_VN_DrawCircle(Exp) [884]
- F_VN_DrawCircles(Exp) [886]
- F_VN_DrawCircularArc(Exp) [888]
- F_VN_FillCircle [917]

#### Contours

- F_VN_DrawContours(Exp) [890]
- F_VN_FillContours [918]

#### Ellipse

- F_VN_DrawEllipse(Exp) [894]
- F_VN_FillEllipse [919]

#### Keypoints and Matches

- F_VN_DrawKeypoints(Exp) [896]
- F_VN_DrawMatches(Exp) [907]

#### Lines

- F_VN_DrawLine(Exp) [898]
- F_VN_DrawLines(Exp) [904]
- F_VN_DrawLine_TcVnVector4_DINT [899]
- F_VN_DrawLineExp_TcVnVector4_LREAL [902]

#### Points

- F_VN_DrawPoint(Exp) [909]
- F_VN_DrawPoints(Exp) [911]

#### Rectangle

- F_VN_DrawRectangle [913]
- F_VN_DrawRectangle_TcVnRectangle_UDINT [914]
• F_VN_DrawRotatedRectangle(Exp) [915]
• F_VN_FillRectangle [920]
• F_VN_FillRotatedRectangle [921]

Text
• F_VN_PutLabel(Exp) [922]
• F_VN_PutText(Exp) [924]

Painting and filling
For many shapes there is an option to paint both the outline of the shape and the filled shape. Two functions are available for this: F_VN_Draw and F_VN_Fill respectively. In addition, setting the outline thickness nThickness to -1 leads to the shape being filled.

6.4.10.1 F_VN_DrawCircle

Draws a circle.

Syntax

Definition:
FUNCTION F_VN_DrawCircle : HRESULT
VAR_INPUT
  nCenterX : UDINT;
  nCenterY : UDINT;
  nRadius : UDINT;
  ipDestImage : ITCvVImage;
  aColor : Reference To TcVnVector4_LREAL;
  nThickness : DINT;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nCenterX</td>
<td>UDINT</td>
<td>x coordinate of the center</td>
</tr>
<tr>
<td>nCenterY</td>
<td>UDINT</td>
<td>y coordinate of the center</td>
</tr>
<tr>
<td>nRadius</td>
<td>UDINT</td>
<td>Radius</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITCvVImage [383]</td>
<td>Destination image</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Color</td>
</tr>
<tr>
<td>nThickness</td>
<td>DINT</td>
<td>Line thickness (if negative, the circle is filled)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
6.4.10.2 F_VN_DrawCircleExp

```
FUNCTION F_VN_DrawCircleExp : HRESULT
VAR_INPUT
    nCenterX : UDINT;
    nCenterY : UDINT;
    nRadius  : UDINT;
    ipDestImage : ITcVnImage;
    aColor : Reference To TcVnVector4_LREAL;
    nThickness : DINT;
    eLineType : ETCvLineType;
    hrPrev : HRESULT;
END_VAR
```

Draws a circle.

Syntax

Definition:

```
FUNCTION F_VN_DrawCircleExp : HRESULT
VAR_INPUT
    nCenterX : UDINT;
    nCenterY : UDINT;
    nRadius  : UDINT;
    ipDestImage : ITcVnImage;
    aColor : Reference To TcVnVector4_LREAL;
    nThickness : DINT;
    eLineType : ETCvLineType;
    hrPrev : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nCenterX</td>
<td>UDINT</td>
<td>x coordinate of the center</td>
</tr>
<tr>
<td>nCenterY</td>
<td>UDINT</td>
<td>y coordinate of the center</td>
</tr>
<tr>
<td>nRadius</td>
<td>UDINT</td>
<td>Radius</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITcVnImage</td>
<td>Destination image</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>Color</td>
</tr>
<tr>
<td>nThickness</td>
<td>DINT</td>
<td>Line thickness (if negative, the circle is filled)</td>
</tr>
<tr>
<td>eLineType</td>
<td>ETCvLineType</td>
<td>Line type</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.10.3 F_VN_DrawCircles

Draws circles.

Syntax

Definition:

FUNCTION F_VN_DrawCircles : HRESULT
VAR_INPUT
  ipCircles : ITCvVnContainer;
  nCircleIndex : DINT;
  ipDestImage : ITCvVnImage;
  aColor : Reference To TcVnVector4_LREAL;
  nThickness : DINT;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipCircles</td>
<td>ITCvVnContainer [345]</td>
<td>Container with circles (ContentType_Vector_TcVnVector3_REAL; Each container element contains the x coordinate of the circle center [0], the y coordinate of the circle center [1], and the radius [2].)</td>
</tr>
<tr>
<td>nCircleIndex</td>
<td>DINT</td>
<td>Index of a specific circle to be drawn (if negative, all circles within the container are drawn)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITCvVnImage [383]</td>
<td>Destination image</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Color</td>
</tr>
<tr>
<td>nThickness</td>
<td>DINT</td>
<td>Line thickness (if negative, the circle is filled)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.10.4 F_VN_DrawCirclesExp

Draws circles. (expert function)

Syntax

Definition:

```c
FUNCTION F_VN_DrawCirclesExp : HRESULT
VAR_INPUT
    ipCircles : ITcVnContainer;
    nCircleIndex : DINT;
    ipDestImage : ITcVnImage;
    aColor : Reference To TcVnVector4_LREAL;
    nThickness : DINT;
    eLineType : ETcVnLineType;
    hrPrev : HRESULT;
END_VAR
```
# Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipCircles</td>
<td>ITcVnContainer [345]</td>
<td>Container with circles (ContainerType_Vector_TcVnVector3_REAL; Each container element contains the x coordinate of the circle center [0], the y coordinate of the circle center [1], and the radius [2].)</td>
</tr>
<tr>
<td>nCircleIndex</td>
<td>DINT</td>
<td>Index of a specific circle to be drawn (if negative, all circles within the container are drawn)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITcVnImage [383]</td>
<td>Destination image</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Color</td>
</tr>
<tr>
<td>nThickness</td>
<td>DINT</td>
<td>Line thickness (if negative, the circle is filled)</td>
</tr>
<tr>
<td>eLineType</td>
<td>ETcVnLineType [194]</td>
<td>Line type</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

# Return value

HRESULT [135]

# Required License

TC3 Vision Base

# System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.10.5 F_VN_DrawCircularArc

```plaintext
F_VN_DrawCircularArc
```

<table>
<thead>
<tr>
<th>stCircularArc</th>
<th>Reference To TcVnCircularArc</th>
<th>HRESULT</th>
<th>F_VN_DrawCircularArc</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipDestImage</td>
<td>ITcVnImage</td>
<td></td>
<td></td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nThickness</td>
<td>DINT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Draws a circular arc.

## Syntax

**Definition:**

```plaintext
FUNCTION F_VN_DrawCircularArc : HRESULT
VAR_INPUT
    stCircularArc : Reference To TcVnCircularArc;
    ipDestImage   : ITcVnImage;
    aColor        : Reference To TcVnVector4_LREAL;
    nThickness    : DINT;
    hrPrev        : HRESULT;
END_VAR
```

888 Version: 1.3 TF7000 - TF7300
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stCircularArc</td>
<td>Reference To TcVnCircularArc</td>
<td>Circular arc definition</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITcVnImage</td>
<td>Destination image</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>Color</td>
</tr>
<tr>
<td>nThickness</td>
<td>DINT</td>
<td>Line thickness (if negative, the arc is filled)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.10.6 F_VN_DrawCircularArcExp

Draws a circular arc. (expert function)

### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_DrawCircularArcExp : HRESULT
VAR_INPUT
   stCircularArc : Reference To TcVnCircularArc;
   ipDestImage : ITcVnImage;
   aColor : Reference To TcVnVector4_LREAL;
   nThickness : DINT;
   eLineType : ETcVnLineType;
   hrPrev : HRESULT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stCircularArc</td>
<td>Reference To TcVnCircularArc</td>
<td>Circular arc definition</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITcVnImage</td>
<td>Destination image</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>Color</td>
</tr>
<tr>
<td>nThickness</td>
<td>DINT</td>
<td>Line thickness (if negative, the arc is filled)</td>
</tr>
<tr>
<td>eLineType</td>
<td>ETcVnLineType</td>
<td>Line type</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.10.7 F_VN_DrawContours

`F_VN_DrawContours` draws a single point set or multiple point sets that are interpreted as contours.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_DrawContours : HRESULT
VAR_INPUT
    ipContours : ITcVnContainer;
    nContourIndex : DINT;
    ipDestImage : ITcVnImage;
    aColor : Reference To TcVnVector4_LREAL;
    nThickness : DINT;
    hrPrev : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContours</td>
<td>ITcVnContainer [345]</td>
<td>Single contour (ContainerType_Vector_TcVnPoint2_DINT) or multiple contours (ContainerType_Vector_Vector_TcVnPoint2_DINT)</td>
</tr>
<tr>
<td>nContourIndex</td>
<td>DINT</td>
<td>Index of a specific contour to be drawn (if negative, all contours within the container are drawn)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITcVnImage [383]</td>
<td>Destination image</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Color</td>
</tr>
<tr>
<td>nThickness</td>
<td>DINT</td>
<td>Line thickness (if negative, the contours are filled)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Further information

The function `F_VN_DrawContours` draws contours in the destination image. A contour is described based on a set of points that are stored in a container. Please note that only integer contour points can be drawn, so that the container must be of type `ContainerType_Vector_TcVnPoint2_DINT`. If the points are described using floating point numbers, `F_VN_ConvertContainerType [688]` can be used to convert the points.

In addition to a container with one contour, a container with several contours can be transferred to the function. This is of type `ContainerType_Vector_Vector_TcVnPoint2_DINT`. The input variable `nContourIndex` indicates which contour is to be drawn. If all contours are to be drawn, `nContourIndex` must be set to a negative value. It is more efficient to draw an entire container of contours at once (set `nContourIndex` to a negative value) than to draw them one after the other by repeatedly calling the function within a loop.

## Expert parameters


## Application

The painting of all contours in a contour list `ipContours` with a red line of 5 px in thickness looks like this, for example:

```plaintext
hr := F_VN_DrawContours(
   ipContours := ipContours,
   nContourIndex := -1,
   ipDestImage := ipImageRes,
   aColor := aColorRed,
   nThickness := 5,
   hrPrev := hr
);
```

### Typical functions that return contours of type `ContainerType_Vector_Vector_TcVnPoint2_DINT`

- `F_VN_DetectBlobs [1029]`
- `F_VN_FindContours [1038]` or `F_VN_FindContoursExp [1041]`
- `F_VN_FindContourHierarchyExp [1034]`

### Samples

- `Blob Detection with watchdog monitoring [1390]`
6.4.10.8 F_VN_DrawContoursExp

F_VN_DrawContoursExp

ipContours ITCvNContainer
nContourIndex DINT
ipDestImage ITCvNImage
aColor Reference To TcVnVector4_LREAL
nThickness DINT
eLineType ETcVnLineType
ipHierarchy ITCvNContainer
nMaxLevel DINT
aOffset Reference To TcVnPoint
hrPrev HRESULT

Draws a single point set or multiple point sets that are interpreted as contours. (expert function)

Syntax

Definition:

FUNCTION F_VN_DrawContoursExp : HRESULT
VAR_INPUT
    ipContours : ITCvNContainer;
    nContourIndex : DINT;
    ipDestImage : ITCvNImage;
    aColor : Reference To TcVnVector4_LREAL;
    nThickness : DINT;
    eLineType : ETcVnLineType;
    ipHierarchy : ITCvNContainer;
    nMaxLevel : DINT;
    aOffset : Reference To TcVnPoint;
    hrPrev : HRESULT;
END_VAR
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContours</td>
<td>ITCvNContainer [345]</td>
<td>Single contour (ContainerType_Vector_TcVnPoint2_DINT) or multiple contours (ContainerType_Vector_Vector_TcVnPoint2_DINT)</td>
</tr>
<tr>
<td>nContourIndex</td>
<td>DINT</td>
<td>Index of a specific contour to be drawn (if negative, all contours within the container are drawn)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITCvNImage [383]</td>
<td>Destination image</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Color</td>
</tr>
<tr>
<td>nThickness</td>
<td>DINT</td>
<td>Line thickness (if negative, the contours are filled)</td>
</tr>
<tr>
<td>eLineType</td>
<td>ETCvNLineType [194]</td>
<td>Line type</td>
</tr>
<tr>
<td>ipHierarchy</td>
<td>ITCvNContainer [345]</td>
<td>Contour hierarchy (ContainerType_Vector_TcVnVector4_DINT)</td>
</tr>
<tr>
<td>nMaxLevel</td>
<td>DINT</td>
<td>Maximum level of contours to be drawn</td>
</tr>
<tr>
<td>aOffset</td>
<td>Reference To TcVnPoint [151]</td>
<td>Offset by which every contour point is shifted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Further information

The function F_VN_DrawContoursExp corresponds to the function F_VN_DrawContours [890], extended by additional input variables.

Thus, the hierarchy description, which is returned by the function F_VN_FindContourHierarchyExp [1033] in addition to the found contours, can also be taken into account. The hierarchy level up to which the contours are to be drawn is specified via ipHierarchy and via the input variable nMaxLevel. If 0 is transferred for ipHierarchy, the hierarchy is ignored.

aOffset can be used to draw all contour points with a uniform offset in x and y direction. This is of interest, for example, if the contours were searched in a ROI but are to be drawn into the original image.

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.10.9  F_VN_DrawEllipse

Defines a function that draws an ellipse.

**Syntax**

Definition:

```c
FUNCTION F_VN_DrawEllipse : HRESULT
VAR_INPUT
  stEllipse   : Reference To TcVnRotatedRectangle;
  ipDestImage : ITcVnImage;
  aColor      : Reference To TcVnVector4_LREAL;
  nThickness  : DINT;
  hrPrev      : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stEllipse</td>
<td>Reference To TcVnRotatedRectangle</td>
<td>Ellipse to be drawn (rotation angle in degrees)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITcVnImage</td>
<td>Destination image</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>Color</td>
</tr>
<tr>
<td>nThickness</td>
<td>DINT</td>
<td>Line thickness (if negative, the ellipse is filled)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.10.10  F_VN_DrawEllipseExp

F_VN_DrawEllipseExp

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stEllipse</td>
<td>Reference To TcVnRotatedRectangle</td>
<td>Ellipse to be drawn (rotation angle in degrees)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITCvImage</td>
<td>Destination image</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>Color</td>
</tr>
<tr>
<td>nThickness</td>
<td>DINT</td>
<td>Line thickness (if negative, the ellipse is filled)</td>
</tr>
<tr>
<td>eLineType</td>
<td>ETCvLineType</td>
<td>Line type</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Syntax**

Definition:

FUNCTION F_VN_DrawEllipseExp : HRESULT
VAR_INPUT
    stEllipse : Reference To TcVnRotatedRectangle;
    ipDestImage : ITCvImage;
    aColor : Reference To TcVnVector4_LREAL;
    nThickness : DINT;
    eLineType : ETCvLineType;
    hrPrev : HRESULT;
END_VAR

**Inputs**

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g.</td>
<td>Tc3_Vision</td>
</tr>
<tr>
<td></td>
<td>Intel 4-core Atom CPU</td>
<td></td>
</tr>
</tbody>
</table>

Draws an ellipse. (expert function)
6.4.10.11  F_VN_DrawKeypoints

F_VN_DrawKeypoints

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvNImage</td>
<td>Source image (elements of type USINT)</td>
</tr>
<tr>
<td>ipKeyPoints</td>
<td>ITCvNContainer</td>
<td>Container with the keypoints (ContainerType_Vector_TcVnKeyPoint)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITCvNImage</td>
<td>Destination image (An appropriate image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Syntax

Definition:
FUNCTION F_VN_DrawKeypoints : HRESULT
VAR_INPUT
  ipSrcImage : ITCvNImage;
  ipKeyPoints : ITCvNContainer;
  ipDestImage : Reference To ITCvNImage;
  hrPrev : HRESULT;
END_VAR

Inputs

Required License

TC3 Vision Matching

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.10.12  F_VN_DrawKeypointsExp

F_VN_DrawKeypointsExp

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvNImage</td>
<td></td>
</tr>
<tr>
<td>ipKeyPoints</td>
<td>ITCvNContainer</td>
<td></td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITCvNImage</td>
<td></td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL</td>
<td></td>
</tr>
<tr>
<td>eFlags</td>
<td>ETcVnDrawMatchesFlags</td>
<td></td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td></td>
</tr>
</tbody>
</table>

Draws the keypoints for visualization purpose.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Draws the keypoints for visualization purpose. (expert function)

**Syntax**

**Definition:**

```
FUNCTION F_VN_DrawKeypointsExp : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  ipKeyPoints : ITcVnContainer;
  ipDestImage : Reference To ITcVnImage;
  aColor : Reference To TcVnVector4_LREAL;
  eFlags : ETcVnDrawMatchesFlags;
  hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image (elements of type USINT)</td>
</tr>
<tr>
<td>ipKeyPoints</td>
<td>ITcVnContainer [345]</td>
<td>Container with the keypoints (ContainerType_Vector_TcVnKeyPoint)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (An appropriate image will be created if required.)</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Color to draw the keypoints (for {-1,-1,-1,-1}, a random color is chosen for each point)</td>
</tr>
<tr>
<td>eFlags</td>
<td>ETcVnDrawMatchesFlags [183]</td>
<td>A combination of flags to support overwriting an existing destination image and/or drawing additional (Rich-)Keypoint information (size and orientation)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

`HRESULT [135]`

**Required License**

TC3 Vision Matching

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.10.13  F_VN_DrawLine

**F_VN_DrawLine**

```c
FUNCTION F_VN_DrawLine : HRESULT
VAR_INPUT
  nX1 : UDINT;
  nY1 : UDINT;
  nX2 : UDINT;
  nY2 : UDINT;
  ipDestImage : ITcVnImage;
  aColor : Reference To TcVnVector4_LREAL;
  nThickness : DINT;
  hrPrev : HRESULT;
END_VAR
```

 Draws a line.

**Syntax**

**Definition:**

FUNCTION F_VN_DrawLine : HRESULT
VAR_INPUT
  nX1 : UDINT;
  nY1 : UDINT;
  nX2 : UDINT;
  nY2 : UDINT;
  ipDestImage : ITcVnImage;
  aColor : Reference To TcVnVector4_LREAL;
  nThickness : DINT;
  hrPrev : HRESULT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nX1</td>
<td>UDINT</td>
<td>x coordinate of the start point</td>
</tr>
<tr>
<td>nY1</td>
<td>UDINT</td>
<td>y coordinate of the start point</td>
</tr>
<tr>
<td>nX2</td>
<td>UDINT</td>
<td>x coordinate of the end point</td>
</tr>
<tr>
<td>nY2</td>
<td>UDINT</td>
<td>y coordinate of the end point</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITcVnImage[1265]</td>
<td>Destination image</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL[153]</td>
<td>Color</td>
</tr>
<tr>
<td>nThickness</td>
<td>DINT</td>
<td>Line thickness</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT[135]

**HRESULT**

16#734 OUTOFRANGE  Part of the line lies outside of the image area. Use the function F_VN_ClipLineToBoundary_ITcVnImage[1265] to ensure that the specified line fits in the image area.

**Required License**

TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.10.14  **F_VN_DrawLine_TcVnVector4_DINT**

Draws a line.

**Syntax**

**Definition:**

```
FUNCTION F_VN_DrawLine_TcVnVector4_DINT : HRESULT
VAR_INPUT
    aLine       : Reference To TcVnVector4_DINT;
    ipDestImage : ITcVnImage;
    aColor      : Reference To TcVnVector4_LREAL;
    nThickness  : DINT;
    hrPrev      : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aLine</td>
<td>Reference To TcVnVector4_DINT</td>
<td>The start and end point of a line segment [x1, y1, x2, y2]</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITcVnImage</td>
<td>Destination image</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>Color</td>
</tr>
<tr>
<td>nThickness</td>
<td>DINT</td>
<td>Line thickness</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

**HRESULT [135]**

**HRESULT**

```
16#734 OUTOFREGION
```

Part of the line lies outside of the image area. Use the function F_VN_ClipLineToBoundary ITcVnImage [1265] to ensure that the specified line fits in the image area.

**Required License**

TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.10.15 F_VN_DrawLine_TcVnVector4_LREAL

**Draws a line.**

**Syntax**

**Definition:**

```pascal
FUNCTION F_VN_DrawLine_TcVnVector4_LREAL : HRESULT
VAR_INPUT
  aLine : Reference To TcVnVector4_LREAL;
  ipDestImage : ITcVnImage;
  aColor : Reference To TcVnVector4_LREAL;
  nThickness : DINT;
  hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aLine</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>The first and second element describe the x and y component of the direction vector. The third and fourth element describe the x and y component of the position vector.</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITcVnImage</td>
<td>Destination image</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>Color</td>
</tr>
<tr>
<td>nThickness</td>
<td>DINT</td>
<td>Line thickness</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**NOTE**

**Floating point exceptions**

This function can unnecessarily cause errors if the option **Floating point exceptions** of the executing PLC task [58] is active. Therefore, deactivate this option.

**Required License**

TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.10.16  F_VN_DrawLineExp

F_VN_DrawLineExp

```plaintext
FUNCTION F_VN_DrawLineExp : HRESULT
VAR_INPUT
    nX1         : UDINT;
nY1         : UDINT;
nX2         : UDINT;
nY2         : UDINT;
ipDestImage : ITCvImage;
aColor      : Reference To TcVnVector4_LREAL;
nThickness  : DINT;
eLineType   : ETCvLineType;
nShift      : UDINT;
hrPrev      : HRESULT;
END_VAR
```

Draws a line. (expert function)

Syntax

Definition:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nX1</td>
<td>UDINT</td>
<td>x coordinate of the start point</td>
</tr>
<tr>
<td>nY1</td>
<td>UDINT</td>
<td>y coordinate of the start point</td>
</tr>
<tr>
<td>nX2</td>
<td>UDINT</td>
<td>x coordinate of the end point</td>
</tr>
<tr>
<td>nY2</td>
<td>UDINT</td>
<td>y coordinate of the end point</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITCvImage [383]</td>
<td>Destination image</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Color</td>
</tr>
<tr>
<td>nThickness</td>
<td>DINT</td>
<td>Line thickness</td>
</tr>
<tr>
<td>eLineType</td>
<td>ETCvLineType [194]</td>
<td>Line type</td>
</tr>
<tr>
<td>nShift</td>
<td>UDINT</td>
<td>Fractional bits of the coordinates (bit shift)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
6.4.10.17  F_VN_DrawLineExp_TcVnVector4_DINT

Draws a line. (expert function)

Syntax

Definition:
FUNCTION F_VN_DrawLineExp_TcVnVector4_DINT : HRESULT
VAR_INPUT
   aLine : Reference To TcVnVector4_DINT;
   ipDestImage : ITCvnImage;
   aColor : Reference To TcVnVector4_LREAL;
   nThickness : DINT;
   eLineType : ETcVnLineType;
   nShift : UDINT;
   hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aLine</td>
<td>Reference To TcVnVector4_DINT</td>
<td>The start and end point of a line segment [x1, y1, x2, y2]</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITCvnImage [383]</td>
<td>Destination image</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>Color</td>
</tr>
<tr>
<td>nThickness</td>
<td>DINT</td>
<td>Line thickness</td>
</tr>
<tr>
<td>eLineType</td>
<td>ETcVnLineType [194]</td>
<td>Line type</td>
</tr>
<tr>
<td>nShift</td>
<td>UDINT</td>
<td>Fractional bits of the coordinates (bit shift)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.10.18  F_VN_DrawLineExp_TcVnVector4_LREAL

F_VN_DrawLineExp_TcVnVector4_LREAL

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aLine</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>The first and second element describe the x and y component of the direction vector. The third and fourth element describe the x and y component of the position vector.</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITCvImage [383]</td>
<td>Destination image</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Color</td>
</tr>
<tr>
<td>nThickness</td>
<td>DINT</td>
<td>Line thickness</td>
</tr>
<tr>
<td>eLineType</td>
<td>ETcVnLineType [194]</td>
<td>Line type</td>
</tr>
<tr>
<td>nShift</td>
<td>UDINT</td>
<td>Fractional bits of the coordinates (bit shift)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Draws a line. (expert function)

Syntax

Definition:

FUNCTION F_VN_DrawLineExp_TcVnVector4_LREAL : HRESULT
VAR_INPUT
    aLine : Reference To TcVnVector4_LREAL;
    ipDestImage : ITCvImage;
    aColor : Reference To TcVnVector4_LREAL;
    nThickness : DINT;
    eLineType : ETcVnLineType;
    nShift : UDINT;
    hrPrev : HRESULT;
END_VAR

Inputs
Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.10.19  F_VN_DrawLines

```plaintext
F_VN_DrawLines
```

Draws lines.

Syntax

**Definition:**

```plaintext
FUNCTION F_VN_DrawLines : HRESULT
VAR_INPUT
  ipLines : ITCvnContainer;
  nLineIndex : DINT;
  ipDestImage : ITCvImage;
  aColor : Reference To TcVnVector4_LREAL;
  nThickness : DINT;
  hrPrev : HRESULT;
END_VAR
```

F_ VN_ DrawLines

ipLines   ITCvnContainer       HRESULT F_ VN_ DrawLines
nLineIndex   DINT
ipDestImage  ITCvImage
aColor      Reference To TcVnVector4_LREAL
nThickness  DINT
hrPrev      HRESULT
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipLines</td>
<td>ITcVnContainer [345]</td>
<td>Container with line descriptions (ContainerType_Vector_TcVnVector2_REAL: the distance from the origin [0] in pixels and the rotation angle [1] in radians. ContainerType_Vector_TcVnVector4_LREAL: the first and second element describe the x and y component of a vector collinear to the line. The third and fourth element describe the x and y component of a point on the line. ContainerType_Vector_TcVnVector4_DINT: start and end point [x1, y1, x2, y2])</td>
</tr>
<tr>
<td>nLineIndex</td>
<td>DINT</td>
<td>Index of a specific line to be drawn (if negative, all lines within the container are drawn)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITcVnImage [383]</td>
<td>Destination image</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Color</td>
</tr>
<tr>
<td>nThickness</td>
<td>DINT</td>
<td>Line thickness</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

**HRESULT [135]**

**HRESULT**

16#734 OUTOFFRAME  Part of the line lies outside of the image area. Use the function F_VN_ClipLineToBoundary ITcVnImage [1265] to ensure that the specified line fits in the image area.

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.10.20  F_VN_DrawLinesExp

Draws lines. (expert function)
Syntax

Definition:

FUNCTION F_VN_DrawLinesExp : HRESULT
VAR_INPUT
  ipLines : ITcVnContainer;
  nLineIndex : DINT;
  ipDestImage : ITcVnImage;
  aColor : Reference To TcVnVector4_LREAL;
  nThickness : DINT;
  eLineType : ETcVnLineType;
  nShift : UDINT;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipLines</td>
<td>ITcVnContainer [345]</td>
<td>Container with line descriptions (ContainerType_Vector_TcVnVector2_REAL: the distance from the origin [0] in pixels and the rotation angle [1] in radians. ContainerType_Vector_TcVnVector4_LREAL: the first and second element describe the x and y component of a vector collinear to the line. The third and fourth element describe the x and y component of a point on the line. ContainerType_Vector_TcVnVector4_LREAL: start and end point [x1, y1, x2, y2])</td>
</tr>
<tr>
<td>nLineIndex</td>
<td>DINT</td>
<td>Index of a specific line to be drawn (if negative, all lines within the container are drawn)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITcVnImage [383]</td>
<td>Destination image</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Color</td>
</tr>
<tr>
<td>nThickness</td>
<td>DINT</td>
<td>Line thickness</td>
</tr>
<tr>
<td>eLineType</td>
<td>ETcVnLineType [194]</td>
<td>Line type</td>
</tr>
<tr>
<td>nShift</td>
<td>UDINT</td>
<td>Fractional bits of the coordinates (bit shift)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
### F_VN_DrawMatches

**Syntax**

**Definition:**

```
FUNCTION F_VN_DrawMatches : HRESULT
VAR_INPUT
    ipSrcImage1 : ITcVnImage;
    ipKeyPoints1 : ITcVnContainer;
    ipSrcImage2 : ITcVnImage;
    ipKeyPoints2 : ITcVnContainer;
    ipMatches1To2 : ITcVnContainer;
    ipDestImage : Reference To ITcVnImage;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage1</td>
<td>ITcVnImage [{383}]</td>
<td>Source image 1 (elements of type USINT)</td>
</tr>
<tr>
<td>ipKeyPoints1</td>
<td>ITcVnContainer [{345}]</td>
<td>Container with the keypoints 1 (ContainerType_Vector_TcVnKeyPoint)</td>
</tr>
<tr>
<td>ipSrcImage2</td>
<td>ITcVnImage [{383}]</td>
<td>Source image 2 (elements of type USINT)</td>
</tr>
<tr>
<td>ipKeyPoints2</td>
<td>ITcVnContainer [{345}]</td>
<td>Container with the keypoints 2 (ContainerType_Vector_TcVnKeyPoint)</td>
</tr>
<tr>
<td>ipMatches1To2</td>
<td>ITcVnContainer [{345}]</td>
<td>Container with the descriptor matches (ContainerType_Vector_TcVnDMatch)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [{383}]</td>
<td>Destination image (An appropriate image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [{135}]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

`HRESULT [{135}]`

**Required License**

TC3 Vision Matching

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.10.22  F_VN_DrawMatchesExp

F_VN_DrawMatchesExp

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage1</td>
<td>ITcVnImage</td>
</tr>
<tr>
<td>ipKeyPoints1</td>
<td>ITcVnContainer</td>
</tr>
<tr>
<td>ipSrcImage2</td>
<td>ITcVnImage</td>
</tr>
<tr>
<td>ipKeyPoints2</td>
<td>ITcVnContainer</td>
</tr>
<tr>
<td>ipMatches1To2</td>
<td>ITcVnContainer</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
</tr>
<tr>
<td>aMatchColor</td>
<td>Reference To TcVnVector4_LREAL</td>
</tr>
<tr>
<td>aSingleColor</td>
<td>Reference To TcVnVector4_LREAL</td>
</tr>
<tr>
<td>ipMatchesMask</td>
<td>ITcVnContainer</td>
</tr>
<tr>
<td>eFlags</td>
<td>ETcVnDrawMatchesFlags</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
</tr>
</tbody>
</table>

Draws the keypoints and matches for visualization purpose. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_DrawMatchesExp : HRESULT
VAR_INPUT
   ipSrcImage1 : ITcVnImage;
   ipKeyPoints1 : ITcVnContainer;
   ipSrcImage2 : ITcVnImage;
   ipKeyPoints2 : ITcVnContainer;
   ipMatches1To2 : ITcVnContainer;
   ipDestImage : Reference To ITcVnImage;
   aMatchColor : Reference To TcVnVector4_LREAL;
   aSingleColor : Reference To TcVnVector4_LREAL;
   ipMatchesMask : ITcVnContainer;
   eFlags : ETcVnDrawMatchesFlags;
   hrPrev : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage1</td>
<td>ITcVnImage [383]</td>
<td>Source image 1 (elements of type USINT)</td>
</tr>
<tr>
<td>ipKeyPoints1</td>
<td>ITcVnContainer [345]</td>
<td>Container with the keypoints 1 (ContainerType_Vector_TcVnKeyPoint)</td>
</tr>
<tr>
<td>ipSrcImage2</td>
<td>ITcVnImage [383]</td>
<td>Source image 2 (elements of type USINT)</td>
</tr>
<tr>
<td>ipKeyPoints2</td>
<td>ITcVnContainer [345]</td>
<td>Container with the keypoints 2 (ContainerType_Vector_TcVnKeyPoint)</td>
</tr>
<tr>
<td>ipMatches1To2</td>
<td>ITcVnContainer [345]</td>
<td>Container with the descriptor matches (ContainerType_Vector_TcVnDMatch)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (An appropriate image will be created if required.)</td>
</tr>
<tr>
<td>aMatchColor</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Color to draw the matches (for {-1, -1, -1, -1}, a random color is chosen for each point and line)</td>
</tr>
<tr>
<td>aSingleColor</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Color to draw the single keypoints (for {-1, -1, -1, -1}, a random color is chosen for each point)</td>
</tr>
<tr>
<td>ipMatchesMask</td>
<td>ITcVnContainer [345]</td>
<td>Mask to select the matches to be drawn (ContainerType_Vector_SINT; Set to 0 if all matches should be drawn.)</td>
</tr>
<tr>
<td>eFlags</td>
<td>ETCVnDrawMatchesFlags [183]</td>
<td>A combination of flags to support overwriting an existing destination image and/or drawing additional (Rich-)Keypoint information (size and orientation) and/or skipping single keypoints</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Matching

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.10.23 F_VN_DrawPoint

```c
F_VN_DrawPoint
```

<table>
<thead>
<tr>
<th>nX UDINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>nY UDINT</td>
</tr>
<tr>
<td>ipDestImage ITcVnImage</td>
</tr>
<tr>
<td>eShape ETCVnDrawShape</td>
</tr>
<tr>
<td>aColor Reference To TcVnVector4_LREAL</td>
</tr>
<tr>
<td>hrPrev HRESULT</td>
</tr>
</tbody>
</table>

Draws a point.
Syntax

Definition:

```c
FUNCTION F_VN_DrawPoint : HRESULT
VAR_INPUT
    nX : UDINT;
    nY : UDINT;
    ipDestImage : ITcVnImage;
    eShape : ETcVnDrawShape;
    aColor : Reference To TcVnVector4_LREAL;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nX</td>
<td>UDINT</td>
<td>x coordinate of the point</td>
</tr>
<tr>
<td>nY</td>
<td>UDINT</td>
<td>y coordinate of the point</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITcVnImage</td>
<td>Destination image</td>
</tr>
<tr>
<td>eShape</td>
<td>ETcVnDrawShape</td>
<td>Point shape</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>Color</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

`HRESULT [135]`

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.10.24** **F_VN_DrawPointExp**

```c
F_VN_DrawPointExp
```

Draw a point. (expert function)
Syntax

Definition:

```
FUNCTION F_VN_DrawPointExp : HRESULT
VAR_INPUT
    nX : UDINT;
    nY : UDINT;
    ipDestImage : ITCvNImage;
    eShape : ETcVnDrawShape;
    aColor : Reference To TcVnVector4_LREAL;
    nSize : UDINT;
    nThickness : DINT;
    eLineType : ETcVnLineType;
    hrPrev : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nX</td>
<td>UDINT</td>
<td>x coordinate of the point</td>
</tr>
<tr>
<td>nY</td>
<td>UDINT</td>
<td>y coordinate of the point</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITCvNImage [138]</td>
<td>Destination image</td>
</tr>
<tr>
<td>eShape</td>
<td>ETcVnDrawShape [184]</td>
<td>Point shape</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Color</td>
</tr>
<tr>
<td>nSize</td>
<td>UDINT</td>
<td>Size of the shape (half width)</td>
</tr>
<tr>
<td>nThickness</td>
<td>DINT</td>
<td>Line thickness (if negative, the shape is filled if it is closed)</td>
</tr>
<tr>
<td>eLineType</td>
<td>ETcVnLineType [194]</td>
<td>Line type</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.10.25  F_VN_DrawPoints

```
F_VN_DrawPoints
    ipPoints ITCvNContainer
    ipDestImage ITCvNImage
    eShape ETcVnDrawShape
    aColor Reference To TcVnVector4_LREAL
    hrPrev HRESULT
RETURN HRESULT F_VN_DrawPoints
```

Draw a collection of points.
API reference

Syntax

Definition:

FUNCTION F_VN_DrawPoints : HRESULT
VAR_INPUT
  ipPoints : ITcVnContainer;
  ipDestImage : ITcVnImage;
  eShape : ETcVnDrawShape;
  aColor : Reference To TcVnVector4_LREAL;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipPoints</td>
<td>ITCvVnContainer [345]</td>
<td>Container with TcVnPoint2_DINT or TcVnPoint2_REAL elements</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITCvVnImage [383]</td>
<td>Destination image</td>
</tr>
<tr>
<td>eShape</td>
<td>ETcVnDrawShape [184]</td>
<td>Point shape</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Color</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.10.26  F_VN_DrawPointsExp

<table>
<thead>
<tr>
<th>ipPoints</th>
<th>ITCvVnContainer</th>
<th>HRESULT</th>
<th>F_VN_DrawPointsExp</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipDestImage</td>
<td>ITCvVnImage</td>
<td></td>
<td></td>
</tr>
<tr>
<td>eShape</td>
<td>ETcVnDrawShape</td>
<td></td>
<td></td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nSize</td>
<td>UDINT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nThickness</td>
<td>DINT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>eLineType</td>
<td>ETcVnLineType</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Draw a collection of points. (expert function)

Syntax

Definition:
FUNCTION F_VN_DrawPointsExp : HRESULT
VAR_INPUT
  ipPoints    : ITcVnContainer;
  ipDestImage : ITcVnImage;
  eShape      : ETcVnDrawShape;
  aColor      : Reference To TcVnVector4_LREAL;
  nSize       : UDINT;
  nThickness  : DINT;
  eLineType   : ETcVnLineType;
  hrPrev      : HRESULT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipPoints</td>
<td>ITcVnContainer [345]</td>
<td>Container with TcVnPoint2_DINT or TcVnPoint2_REAL elements</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITcVnImage [383]</td>
<td>Destination image</td>
</tr>
<tr>
<td>eShape</td>
<td>ETcVnDrawShape [184]</td>
<td>Point shape</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Color</td>
</tr>
<tr>
<td>nSize</td>
<td>UDINT</td>
<td>Size of the shape (half width)</td>
</tr>
<tr>
<td>nThickness</td>
<td>DINT</td>
<td>Line thickness (if negative, the shape is filled if it is closed)</td>
</tr>
<tr>
<td>eLineType</td>
<td>ETcVnLineType [194]</td>
<td>Line type</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.10.27 F_VN_DrawRectangle**

```
F_VN_DrawRectangle
nTopLeftX UDINT
nTopLeftY UDINT
nBottomRightX UDINT
nBottomRightY UDINT
ipDestImage ITcVnImage
aColor Reference To TcVnVector4_LREAL
nThickness DINT
hrPrev HRESULT
```

Draw a rectangle.
API reference

Syntax

Definition:

FUNCTION F_VN_DrawRectangle : HRESULT
VAR_INPUT
    nTopLeftX : UDINT;
    nTopLeftY : UDINT;
    nBottomRightX : UDINT;
    nBottomRightY : UDINT;
    ipDestImage : ITcVnImage;
    aColor : Reference To TcVnVector4_LREAL;
    nThickness : DINT;
    hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nTopLeftX</td>
<td>UDINT</td>
<td>x coordinate of the top left corner</td>
</tr>
<tr>
<td>nTopLeftY</td>
<td>UDINT</td>
<td>y coordinate of the top left corner</td>
</tr>
<tr>
<td>nBottomRightX</td>
<td>UDINT</td>
<td>x coordinate of the bottom right corner</td>
</tr>
<tr>
<td>nBottomRightY</td>
<td>UDINT</td>
<td>y coordinate of the bottom right corner</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITcVnImage[383]</td>
<td>Destination image</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL[153]</td>
<td>Color</td>
</tr>
<tr>
<td>nThickness</td>
<td>DINT</td>
<td>Line thickness (if negative, the rectangle is filled)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT[135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.10.28 F_VN_DrawRectangle_TcVnRectangle_UDINT

F_VN_DrawRectangle_TcVnRectangle_UDINT

<table>
<thead>
<tr>
<th>stRectangle</th>
<th>Reference To TcVnRectangle_UDINT</th>
<th>HRESULT F_VN_DrawRectangle_TcVnRectangle_UDINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipDestImage</td>
<td>ITcVnImage</td>
<td></td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL</td>
<td></td>
</tr>
<tr>
<td>nThickness</td>
<td>DINT</td>
<td></td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td></td>
</tr>
</tbody>
</table>

Draw a rectangle.

Syntax

Definition:
FUNCTION F_VN_DrawRectangle_TcVnRectangle_UDINT : HRESULT
VAR_INPUT
    stRectangle : Reference To TcVnRectangle_UDINT;
    ipDestImage : ITcVnImage;
    aColor : Reference To TcVnVector4_LREAL;
    nThickness : DINT;
    hrPrev : HRESULT;
END_VAR

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stRectangle</td>
<td>Reference To TcVnRectangle_UDINT [222]</td>
<td>Rectangle to be drawn</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITcVnImage [383]</td>
<td>Destination image</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Color</td>
</tr>
<tr>
<td>nThickness</td>
<td>DINT</td>
<td>Line thickness (if negative, the rectangle is filled)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.10.29 F_VN_DrawRotatedRectangle

```c
FUNCTION F_VN_DrawRotatedRectangle : HRESULT
VAR_INPUT
    stRectangle : Reference To TcVnRotatedRectangle;
    ipDestImage : ITcVnImage;
    aColor : Reference To TcVnVector4_LREAL;
    nThickness : DINT;
    hrPrev : HRESULT;
END_VAR
```

Draw a rectangle.

### Syntax

**Definition:**

FUNCTION F_VN_DrawRotatedRectangle : HRESULT

VAR_INPUT
    stRectangle : Reference To TcVnRotatedRectangle;
    ipDestImage : ITcVnImage;
    aColor : Reference To TcVnVector4_LREAL;
    nThickness : DINT;
    hrPrev : HRESULT;
END_VAR
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stRectangle</td>
<td>Reference To TcVnRotatedRectangle</td>
<td>Rectangle to be drawn (rotation angle in degrees)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITcVnImage</td>
<td>Destination image</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>Color</td>
</tr>
<tr>
<td>nThickness</td>
<td>DINT</td>
<td>Line thickness (if negative, the rectangle is filled)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.10.30 F_VN_DrawRotatedRectangleExp

Draw a rectangle. (expert function)

Syntax

Definition:

FUNCTION F_VN_DrawRotatedRectangleExp : HRESULT
VAR_INPUT
    stRectangle : Reference To TcVnRotatedRectangle;
    ipDestImage : ITcVnImage;
    aColor : Reference To TcVnVector4_LREAL;
    nThickness : DINT;
    eLineType : ETcVnLineType;
    hrPrev : HRESULT;
END_VAR
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stRectangle</td>
<td>Reference To TcVnRotatedRectangle</td>
<td>Rectangle to be drawn (rotation angle in degrees)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITcVnImage</td>
<td>Destination image</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>Color</td>
</tr>
<tr>
<td>nThickness</td>
<td>DINT</td>
<td>Line thickness (if negative, the rectangle is filled)</td>
</tr>
<tr>
<td>eLineType</td>
<td>ETcVnLineType</td>
<td>Line type</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

**HRESULT [135]**

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.10.31 F_VN_FillCircle**

Paint a filled circle.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_FillCircle : HRESULT
VAR_INPUT
  nCenterX    : UDINT;
  nCenterY    : UDINT;
  nRadius     : UDINT;
  ipDestImage : ITcVnImage;
  aColor      : Reference To TcVnVector4_LREAL;
  hrPrev      : HRESULT;
END_VAR
```

F_VN_FillCircle

nCenterX UDINT
nCenterY UDINT
nRadius UDINT
ipDestImage ITcVnImage
aColor Reference To TcVnVector4_LREAL
hrPrev HRESULT
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nCenterX</td>
<td>UDINT</td>
<td>x coordinate of the center</td>
</tr>
<tr>
<td>nCenterY</td>
<td>UDINT</td>
<td>y coordinate of the center</td>
</tr>
<tr>
<td>nRadius</td>
<td>UDINT</td>
<td>Radius</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITcVnImage</td>
<td>Destination image</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>Color</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## Syntax

**F_VN_FillContours**

```
FUNCTION F_VN_FillContours : HRESULT
VAR_INPUT
    ipContours       : ITcVnContainer;
    nContourIndex    : DINT;
    ipDestImage      : ITcVnImage;
    aColor           : Reference To TcVnVector4_LREAL;
    hrPrev           : HRESULT;
END_VAR
```

Paint a single point set or multiple point sets that are interpreted as contours.

## 6.4.10.32  F_VN_FillContours

Paint a single point set or multiple point sets that are interpreted as contours.
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContours</td>
<td>ITCvNContainer</td>
<td>Single contour (ContainerType_Vector_TcVnPoint2_DINT) or multiple contours (ContainerType_Vector_Vector_TcVnPoint2_DINT)</td>
</tr>
<tr>
<td>nContourIndex</td>
<td>DINT</td>
<td>Index of a specific contour to be drawn (if negative, all contours within the container are drawn)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITCvNImage</td>
<td>Destination image</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>Color</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.10.33 F_VN_FillEllipse

```vbnet
FUNCTION F_VN_FillEllipse : HRESULT
VAR_INPUT
    stEllipse : Reference To TcVnRotatedRectangle;
    ipDestImage : ITCvNImage;
    aColor : Reference To TcVnVector4_LREAL;
    hrPrev : HRESULT;
END_VAR
```

Paint a filled ellipse.

## Syntax

### Definition:

```vbnet
FUNCTION F_VN_FillEllipse : HRESULT
```

```vbnet
VAR_INPUT
    stEllipse : Reference To TcVnRotatedRectangle;
    ipDestImage : ITCvNImage;
    aColor : Reference To TcVnVector4_LREAL;
    hrPrev : HRESULT;
END_VAR
```
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stEllipse</td>
<td>Reference To TcVnRotatedRectangle[222]</td>
<td>Ellipse to be painted (rotation angle in degrees)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITcVnImage[383]</td>
<td>Destination image</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL[153]</td>
<td>Color</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT[135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.10.34 F_VN_FillRectangle

Paint a filled rectangle.

#### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_FillRectangle : HRESULT
VAR_INPUT
  nTopLeftX : UDINT;
  nTopLeftY : UDINT;
  nBottomRightX : UDINT;
  nBottomRightY : UDINT;
  ipDestImage : ITcVnImage;
  aColor : Reference To TcVnVector4_LREAL;
  hrPrev : HRESULT;
END_VAR
```
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nTopLeftX</td>
<td>UDINT</td>
<td>x coordinate of the top left corner</td>
</tr>
<tr>
<td>nTopLeftY</td>
<td>UDINT</td>
<td>y coordinate of the top left corner</td>
</tr>
<tr>
<td>nBottomRightX</td>
<td>UDINT</td>
<td>x coordinate of the bottom right corner</td>
</tr>
<tr>
<td>nBottomRightY</td>
<td>UDINT</td>
<td>y coordinate of the bottom right corner</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITcVnImage</td>
<td>Destination image</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>Color</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.10.35 F_VN_FillRotatedRectangle

Paint a filled rectangle.

Syntax

Definition:

FUNCTION F_VN_FillRotatedRectangle : HRESULT
VAR_INPUT
  stRectangle : Reference To TcVnRotatedRectangle;
  ipDestImage : ITcVnImage;
  aColor : Reference To TcVnVector4_LREAL;
  hrPrev : HRESULT;
END_VAR
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stRectangle</td>
<td>Reference To TcVnRotatedRectangle</td>
<td>Rectangle to be painted (rotation angle in degrees)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITcVnImage</td>
<td>Destination image</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>Color</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.10.36 F_VN_PutLabel

Write a label (black text on white background) into an image.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_PutLabel : HRESULT
VAR_INPUT
sText : STRING;
ipDestImage : ITcVnImage;
nX : UDINT;
nY : UDINT;
fFontScale : LREAL;
hrPrev : HRESULT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sText</td>
<td>STRING</td>
<td>Text</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>TcVnImage</td>
<td>Destination image</td>
</tr>
<tr>
<td>nX</td>
<td>UDINT</td>
<td>x coordinate (bottom left)</td>
</tr>
<tr>
<td>nY</td>
<td>UDINT</td>
<td>y coordinate (bottom left)</td>
</tr>
<tr>
<td>fFontScale</td>
<td>LREAL</td>
<td>Scaling factor</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.10.37 F_VN_PutLabelExp**

Write a label (text on unified background) into an image. (expert function)

**Syntax**

Definition:

```plaintext
FUNCTION F_VN_PutLabelExp : HRESULT
VAR_INPUT
  sText : STRING;
  ipDestImage : TcVnImage;
  nX : UDINT;
  nY : UDINT;
  fFontScale : LREAL;
  nThickness : UDINT;
  eFontType : ETcVnFontType;
  aFontColor : Reference To TcVnVector4_LREAL;
  aBackgroundColor : Reference To TcVnVector4_LREAL;
  eLineType : ETcVnLineType;
  hrPrev : HRESULT;
```

F_VN_PutLabelExp

Write a label (text on unified background) into an image. (expert function)
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sText</td>
<td>STRING</td>
<td>Text</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITcVnImage [➔ 383]</td>
<td>Destination image</td>
</tr>
<tr>
<td>nX</td>
<td>UDINT</td>
<td>x coordinate (bottom left)</td>
</tr>
<tr>
<td>nY</td>
<td>UDINT</td>
<td>y coordinate (bottom left)</td>
</tr>
<tr>
<td>fFontScale</td>
<td>LREAL</td>
<td>Scaling factor</td>
</tr>
<tr>
<td>nThickness</td>
<td>UDINT</td>
<td>Line thickness</td>
</tr>
<tr>
<td>eFontType</td>
<td>ETcVnFontType [➔ 190]</td>
<td>Font type</td>
</tr>
<tr>
<td>aFontColor</td>
<td>Reference To TcVnVector4_LREAL [➔ 153]</td>
<td>Font color</td>
</tr>
<tr>
<td>aBackgroundColor</td>
<td>Reference To TcVnVector4_LREAL [➔ 153]</td>
<td>Background color</td>
</tr>
<tr>
<td>eLineType</td>
<td>ETcVnLineType [➔ 194]</td>
<td>Line type</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [➔ 135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [➔ 135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.10.38 F_VN_PutText

Write text into an image.
Syntax

Definition:

FUNCTION F_VN_PutText : HRESULT
VAR_INPUT
  sText : STRING;
  ipDestImage : ITCvNImage;
  nX : UDINT;
  nY : UDINT;
  eFontType : ETcVnFontType;
  fFontScale : LREAL;
  aColor : Reference To TcVnVector4_LREAL;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sText</td>
<td>STRING</td>
<td>Text</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITCvNImage [383]</td>
<td>Destination image</td>
</tr>
<tr>
<td>nX</td>
<td>UDINT</td>
<td>x coordinate (bottom left)</td>
</tr>
<tr>
<td>nY</td>
<td>UDINT</td>
<td>y coordinate (bottom left)</td>
</tr>
<tr>
<td>eFontType</td>
<td>ETcVnFontType [190]</td>
<td>Font type</td>
</tr>
<tr>
<td>fFontScale</td>
<td>LREAL</td>
<td>Scaling factor</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Text color</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Further information

The following fonts are available:
Hershey Simplex
  Hershey Plain
Hershey Duplex
Hershey Complex
Hershey Triplex
Hershey Complex Small
Hershey Script Simplex
Hershey Script Complex
  Plain Italic
 Complex Italic
 Triplex Italic
 Complex Small Italic

Related functions
  • F_VN_PutLabelExp [ 923 ]
  • F_VN_PutTextExp [ 924 ]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.10.39  F_VN_PutTextExp

```
F_VN_PutTextExp

sText      STRING
ipDestImage ITCVnImage
nX          UDINT
nY          UDINT
eFontType   ETCVnFontType
fFontScale  LREAL
aColor      Reference To TcVnVector4_LREAL
nThickness  DINT
eLineType   ETCVnLineType
bBottomLeftOrigin BOOL
hrPrev      HRESULT
```

Write text into an image. (expert function)
Syntax

Definition:

FUNCTION F_VN_PutTextExp : HRESULT
VAR_INPUT
  sText : STRING;
  ipDestImage : ITcVnImage;
  nX : UDINT;
  nY : UDINT;
  eFontType : ETcVnFontType;
  fFontScale : LREAL;
  aColor : Reference To TcVnVector4_LREAL;
  nThickness : DINT;
  eLineType : ETcVnLineType;
  bBottomLeftOrigin : BOOL;
hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sText</td>
<td>STRING</td>
<td>Text</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>ITcVnImage</td>
<td>Destination image</td>
</tr>
<tr>
<td>nX</td>
<td>UDINT</td>
<td>x coordinate (bottom left)</td>
</tr>
<tr>
<td>nY</td>
<td>UDINT</td>
<td>y coordinate (bottom left)</td>
</tr>
<tr>
<td>eFontType</td>
<td>ETcVnFontType</td>
<td>Font type</td>
</tr>
<tr>
<td>fFontScale</td>
<td>LREAL</td>
<td>Scaling factor</td>
</tr>
<tr>
<td>aColor</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>Text color</td>
</tr>
<tr>
<td>nThickness</td>
<td>DINT</td>
<td>Line thickness</td>
</tr>
<tr>
<td>eLineType</td>
<td>ETcVnLineType</td>
<td>Line type</td>
</tr>
<tr>
<td>bBottomLeftOrigin</td>
<td>BOOL</td>
<td>Sets the image origin to the bottom left corner, if true</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.11 Fourier Analysis

This group contains functions for frequency analysis and frequency filtering in images.
Functions

Butterworth Filter

- F_VN_CreateBandpassButterworthFilter [928]
- F_VN_CreateBandrejectButterworthFilter [930]
- F_VN_CreateHighpassButterworthFilter [933]
- F_VN_CreateLowpassButterworthFilter [935]

Gaussian filter

- F_VN_CreateBandpassGaussianFilter [929]
- F_VN_CreateBandrejectGaussianFilter [932]
- F_VN_CreateHighpassGaussianFilter [934]
- F_VN_CreateLowpassGaussianFilter [936]

Discrete Fourier Transformation

- F_VN_Dft [937]
- F_VN_InverseDft [938]
- F_VN_OptimalDftSize [939]

Miscellaneous

- F_VN_PadImageBorder(Exp) [940]

6.4.11.1 F_VN_CreateBandpassButterworthFilter

F_VN_CreateBandpassButterworthFilter

ipFilter Reference To ITcVnImage
nWidth UDINT
nHeight UDINT
bDoublePrecision BOOL
bOriginAtCenter BOOL
fCutoffDistance LREAL
fBandWidth LREAL
nOrder UDINT
fScale LREAL
hrPrev HRESULT

Creates a bandpass Butterworth filter, which can be applied to a frequency domain image by element-wise multiplication.

Syntax

Definition:

FUNCTION F_VN_CreateBandpassButterworthFilter : HRESULT
VAR_INPUT
  ipFilter : Reference To ITcVnImage;
  nWidth : UDINT;
  nHeight : UDINT;
  bDoublePrecision : BOOL;
  bOriginAtCenter : BOOL;
  fCutoffDistance : LREAL;
  fBandWidth : LREAL;
  nOrder : UDINT;
  fScale : LREAL;
  hrPrev : HRESULT;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipFilter</td>
<td>Reference To ITcVnImage</td>
<td>Returns the created filter (1 channel of type REAL or LREAL, depending on bDoublePrecision. Non-zero interface pointers are reused.).</td>
</tr>
<tr>
<td>nWidth</td>
<td>UDINT</td>
<td>Filter width (even, &gt;= 2, must match the width of the image it is applied to)</td>
</tr>
<tr>
<td>nHeight</td>
<td>UDINT</td>
<td>Filter height (even, &gt;= 2, must match the height of the image it is applied to)</td>
</tr>
<tr>
<td>bDoublePrecision</td>
<td>BOOL</td>
<td>If true, the filter is generated with double precision (LREAL) instead of single precision (REAL).</td>
</tr>
<tr>
<td>bOriginAtCenter</td>
<td>BOOL</td>
<td>If true, the filter origin is shifted to the image center.</td>
</tr>
<tr>
<td>fCutoffDistance</td>
<td>LREAL</td>
<td>The cutoff distance of the Butterworth filter (&gt; 0).</td>
</tr>
<tr>
<td>fBandWidth</td>
<td>LREAL</td>
<td>The band width of the Butterworth filter (&gt; 0).</td>
</tr>
<tr>
<td>nOrder</td>
<td>UDINT</td>
<td>The order of the Butterworth filter (&gt; 0).</td>
</tr>
<tr>
<td>fScale</td>
<td>LREAL</td>
<td>The scale factor of the Butterworth filter denominator term (&gt; 0, e.g. 0.414 or 1.0).</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.11.2 F_VN_CreateBandpassGaussianFilter

F_VN_CreateBandpassGaussianFilter

ipFilter : Reference To ITcVnImage
nWidth : UDINT
nHeight : UDINT
bDoublePrecision : BOOL
bOriginAtCenter : BOOL
fCutoffDistance : LREAL
fBandWidth : LREAL
hrPrev : HRESULT

Creates a bandpass Gaussian filter, which can be applied to a frequency domain image by element-wise multiplication.

### Syntax

**Definition:**

FUNCTION F_VN_CreateBandpassGaussianFilter : HRESULT
VAR_INPUT
    ipFilter : Reference To ITcVnImage;
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipFilter</td>
<td>Reference To ITcVnImage [383]</td>
<td>Returns the created filter (1 channel of type REAL or LREAL, depending on bDoublePrecision. Non-zero interface pointers are reused.).</td>
</tr>
<tr>
<td>nWidth</td>
<td>UDINT</td>
<td>Filter width (even, &gt;= 2, must match the width of the image it is applied to)</td>
</tr>
<tr>
<td>nHeight</td>
<td>UDINT</td>
<td>Filter height (even, &gt;= 2, must match the height of the image it is applied to)</td>
</tr>
<tr>
<td>bDoublePrecision</td>
<td>BOOL</td>
<td>If true, the filter is generated with double precision (LREAL) instead of single precision (REAL).</td>
</tr>
<tr>
<td>bOriginAtCenter</td>
<td>BOOL</td>
<td>If true, the filter origin is shifted to the image center.</td>
</tr>
<tr>
<td>fCutoffDistance</td>
<td>LREAL</td>
<td>The cutoff distance of the Gaussian filter (&gt; 0).</td>
</tr>
<tr>
<td>fBandWidth</td>
<td>LREAL</td>
<td>The band width of the Gaussian filter (&gt; 0).</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.).</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.11.3 F_VN_CreateBandrejectButterworthFilter

Creates a bandreject Butterworth filter, which can be applied to a frequency domain image by element-wise multiplication.
Syntax

Definition:

```cpp
FUNCTION F_VN_CreateBandrejectButterworthFilter : HRESULT
VAR_INPUT
    ipFilter : Reference To ITcVnImage;
    nWidth : UDINT;
    nHeight : UDINT;
    bDoublePrecision : BOOL;
    bOriginAtCenter : BOOL;
    fCutoffDistance : LREAL;
    fBandWidth : LREAL;
    nOrder : UDINT;
    fScale : LREAL;
    hrPrev : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipFilter</td>
<td>Reference To ITcVnImage</td>
<td>Returns the created filter (1 channel of type REAL or LREAL, depending on bDoublePrecision. Non-zero interface pointers are reused.).</td>
</tr>
<tr>
<td>nWidth</td>
<td>UDINT</td>
<td>Filter width (even, &gt;= 2, must match the width of the image it is applied to)</td>
</tr>
<tr>
<td>nHeight</td>
<td>UDINT</td>
<td>Filter height (even, &gt;= 2, must match the height of the image it is applied to)</td>
</tr>
<tr>
<td>bDoublePrecision</td>
<td>BOOL</td>
<td>If true, the filter is generated with double precision (LREAL) instead of single precision (REAL).</td>
</tr>
<tr>
<td>bOriginAtCenter</td>
<td>BOOL</td>
<td>If true, the filter origin is shifted to the image center.</td>
</tr>
<tr>
<td>fCutoffDistance</td>
<td>LREAL</td>
<td>The cutoff distance of the Butterworth filter (&gt; 0).</td>
</tr>
<tr>
<td>fBandWidth</td>
<td>LREAL</td>
<td>The band width of the Butterworth filter (&gt; 0).</td>
</tr>
<tr>
<td>nOrder</td>
<td>UDINT</td>
<td>The order of the Butterworth filter (&gt; 0).</td>
</tr>
<tr>
<td>fScale</td>
<td>LREAL</td>
<td>The scale factor of the Butterworth filter denominator term (&gt; 0, e.g. 0.414 or 1.0).</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
Creates a bandreject Gaussian filter, which can be applied to a frequency domain image by element-wise multiplication.

**Syntax**

**Definition:**

```
FUNCTION F_VN_CreateBandrejectGaussianFilter : HRESULT
VAR_INPUT
  ipFilter : Reference To ITcVnImage;
  nWidth : UDINT;
  nHeight : UDINT;
  bDoublePrecision : BOOL;
  bOriginAtCenter : BOOL;
  fCutoffDistance : LREAL;
  fBandWidth : LREAL;
  hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipFilter</td>
<td>Reference To ITcVnImage</td>
<td>Returns the created filter (1 channel of type REAL or LREAL, depending on bDoublePrecision. Non-zero interface pointers are reused.).</td>
</tr>
<tr>
<td>nWidth</td>
<td>UDINT</td>
<td>Filter width (even, &gt;= 2, must match the width of the image it is applied to)</td>
</tr>
<tr>
<td>nHeight</td>
<td>UDINT</td>
<td>Filter height (even, &gt;= 2, must match the height of the image it is applied to)</td>
</tr>
<tr>
<td>bDoublePrecision</td>
<td>BOOL</td>
<td>If true, the filter is generated with double precision (LREAL) instead of single precision (REAL).</td>
</tr>
<tr>
<td>bOriginAtCenter</td>
<td>BOOL</td>
<td>If true, the filter origin is shifted to the image center.</td>
</tr>
<tr>
<td>fCutoffDistance</td>
<td>LREAL</td>
<td>The cutoff distance of the Gaussian filter (&gt; 0).</td>
</tr>
<tr>
<td>fBandWidth</td>
<td>LREAL</td>
<td>The band width of the Gaussian filter (&gt; 0).</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

**HRESULT [135]**

**Required License**

TC3 Vision Base
6.4.11.5  F_VN_CreateHighpassButterworthFilter

Creates a highpass Butterworth filter, which can be applied to a frequency domain image by element-wise multiplication.

Syntax

Definition:
FUNCTION F_VN_CreateHighpassButterworthFilter : HRESULT
VAR_INPUT
   ipFilter : Reference To ITcVnImage;
   nWidth : UDINT;
   nHeight : UDINT;
   bDoublePrecision : BOOL;
   bOriginAtCenter : BOOL;
   fCutoffDistance : LREAL;
   nOrder : UDINT;
   fScale : LREAL;
   hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipFilter</td>
<td>Reference To ITcVnImage</td>
<td>Returns the created filter (1 channel of type REAL or LREAL, depending on bDoublePrecision. Non-zero interface pointers are reused.).</td>
</tr>
<tr>
<td>nWidth</td>
<td>UDINT</td>
<td>Filter width (even, &gt;= 2, must match the width of the image it is applied to)</td>
</tr>
<tr>
<td>nHeight</td>
<td>UDINT</td>
<td>Filter height (even, &gt;= 2, must match the height of the image it is applied to)</td>
</tr>
<tr>
<td>bDoublePrecision</td>
<td>BOOL</td>
<td>If true, the filter is generated with double precision (LREAL) instead of single precision (REAL).</td>
</tr>
<tr>
<td>bOriginAtCenter</td>
<td>BOOL</td>
<td>If true, the filter origin is shifted to the image center.</td>
</tr>
<tr>
<td>fCutoffDistance</td>
<td>LREAL</td>
<td>The cutoff distance of the Butterworth filter (&gt; 0).</td>
</tr>
<tr>
<td>nOrder</td>
<td>UDINT</td>
<td>The order of the Butterworth filter (&gt; 0).</td>
</tr>
<tr>
<td>fScale</td>
<td>LREAL</td>
<td>The scale factor of the Butterworth filter denominator term (&gt; 0, e.g. 0.414 or 1.0).</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
6.4.11.6 F_VN_CreateHighpassGaussianFilter

Creates a highpass Gaussian filter, which can be applied to a frequency domain image by element-wise multiplication.

Syntax

Definition:

FUNCTION F_VN_CreateHighpassGaussianFilter : HRESULT

VAR_INPUT

  ipFilter : Reference To ITcVnImage;
  nWidth : UDINT;
  nHeight : UDINT;
  bDoublePrecision : BOOL;
  bOriginAtCenter : BOOL;
  fCutoffDistance : LREAL;
  hrPrev : HRESULT;

END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipFilter</td>
<td>Reference To ITcVnImage</td>
<td>Returns the created filter (1 channel of type REAL or LREAL, depending on bDoublePrecision. Non-zero interface pointers are reused.).</td>
</tr>
<tr>
<td>nWidth</td>
<td>UDINT</td>
<td>Filter width (even, &gt;= 2, must match the width of the image it is applied to)</td>
</tr>
<tr>
<td>nHeight</td>
<td>UDINT</td>
<td>Filter height (even, &gt;= 2, must match the height of the image it is applied to)</td>
</tr>
<tr>
<td>bDoublePrecision</td>
<td>BOOL</td>
<td>If true, the filter is generated with double precision (LREAL) instead of single precision (REAL).</td>
</tr>
<tr>
<td>bOriginAtCenter</td>
<td>BOOL</td>
<td>If true, the filter origin is shifted to the image center.</td>
</tr>
<tr>
<td>fCutoffDistance</td>
<td>LREAL</td>
<td>The cutoff distance of the Gaussian filter (&gt; 0).</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.11.7 F_VN_CreateLowpassButterworthFilter

F_VN_CreateLowpassButterworthFilter

ipFilter Reference To ITcVnImage   HRESULT F_VN_CreateLowpassButterworthFilter
nWidth UDINT
nHeight UDINT
bDoublePrecision BOOL
bOriginAtCenter BOOL
fCutoffDistance LREAL
nOrder UDINT
fScale LREAL
hrPrev HRESULT

Creates a lowpass Butterworth filter, which can be applied to a frequency domain image by element-wise multiplication.

Syntax

Definition:
FUNCTION F_VN_CreateLowpassButterworthFilter : HRESULT
VAR_INPUT
ipFilter : Reference To ITcVnImage;
nWidth : UDINT;
nHeight : UDINT;
bDoublePrecision : BOOL;
bOriginAtCenter : BOOL;
fCutoffDistance : LREAL;
nOrder : UDINT;
fScale : LREAL;
hPrev : HRESULT;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipFilter</td>
<td>Reference To ITcVnImage</td>
<td>Returns the created filter (1 channel of type REAL or LREAL, depending on bDoublePrecision. Non-zero interface pointers are reused.).</td>
</tr>
<tr>
<td>nWidth</td>
<td>UDINT</td>
<td>Filter width (even, &gt;= 2, must match the width of the image it is applied to)</td>
</tr>
<tr>
<td>nHeight</td>
<td>UDINT</td>
<td>Filter height (even, &gt;= 2, must match the height of the image it is applied to)</td>
</tr>
<tr>
<td>bDoublePrecision</td>
<td>BOOL</td>
<td>If true, the filter is generated with double precision (LREAL) instead of single precision (REAL).</td>
</tr>
<tr>
<td>bOriginAtCenter</td>
<td>BOOL</td>
<td>If true, the filter origin is shifted to the image center.</td>
</tr>
<tr>
<td>fCutoffDistance</td>
<td>LREAL</td>
<td>The cutoff distance of the Butterworth filter (&gt; 0).</td>
</tr>
<tr>
<td>nOrder</td>
<td>UDINT</td>
<td>The order of the Butterworth filter (&gt; 0).</td>
</tr>
<tr>
<td>fScale</td>
<td>LREAL</td>
<td>The scale factor of the Butterworth filter denominator term (&gt; 0, e.g. 0.414 or 1.0).</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.11.8 F_VN_CreateLowpassGaussianFilter

F_VN_CreateLowpassGaussianFilter

**ipFilter** Reference To ITcVnImage

**nWidth** UDINT

**nHeight** UDINT

**bDoublePrecision** BOOL

**bOriginAtCenter** BOOL

**fCutoffDistance** LREAL

**hrPrev** HRESULT

Creates a lowpass Gaussian filter, which can be applied to a frequency domain image by element-wise multiplication.

### Syntax

**Definition:**

FUNCTION F_VN_CreateLowpassGaussianFilter : HRESULT

VAR_INPUT

ipFilter : Reference To ITcVnImage;
nWidth : UDINT;
nHeight : UDINT;
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipFilter</td>
<td>Reference To ITcVnImage</td>
<td>Returns the created filter (1 channel of type REAL or LREAL, depending on bDoublePrecision. Non-zero interface pointers are reused.).</td>
</tr>
<tr>
<td>nWidth</td>
<td>UDINT</td>
<td>Filter width (even, &gt;= 2, must match the width of the image it is applied to)</td>
</tr>
<tr>
<td>nHeight</td>
<td>UDINT</td>
<td>Filter height (even, &gt;= 2, must match the height of the image it is applied to)</td>
</tr>
<tr>
<td>bDoublePrecision</td>
<td>BOOL</td>
<td>If true, the filter is generated with double precision (LREAL) instead of single precision (REAL).</td>
</tr>
<tr>
<td>bOriginAtCenter</td>
<td>BOOL</td>
<td>If true, the filter origin is shifted to the image center.</td>
</tr>
<tr>
<td>fCutoffDistance</td>
<td>LREAL</td>
<td>The cutoff distance of the Gaussian filter (&gt; 0).</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64)</td>
<td>Tc3_Vision</td>
</tr>
<tr>
<td></td>
<td>with PL50, e.g.</td>
<td>Intel 4-core Atom CPU</td>
</tr>
</tbody>
</table>

6.4.11.9 F_VN_Dft

Computes the DFT for a given image.

Syntax

Definition:

FUNCTION F_VN_Dft : HRESULT
VAR_INPUT
  ipSpatialImage : ITcVnImage;
  ipFrequencyImage : Reference To ITcVnImage;
  bPackedCCS : BOOL;
  bAutoPadding : BOOL;
  hrPrev : HRESULT;
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSpatialImage</td>
<td>ITCvNImage [383]</td>
<td>Source image (elements of type REAL or LREAL, 1 (Re) or 2 (Re + Im) channels)</td>
</tr>
<tr>
<td>ipFrequencyImage</td>
<td>Reference To ITCvNImage [383]</td>
<td>Destination image (same type as ipSpatialImage, but number of channels and size can vary depending on bPackedCCS and bAutoPadding.</td>
</tr>
<tr>
<td>bPackedCCS</td>
<td>BOOL</td>
<td>If true and ipSpatialImage has only 1 channel, the result image will have 1 channel with packed complex-conjugate-symmetrical format results. Otherwise, the result image will have 2 separate channels (Re + Im), containing the full spectrum.</td>
</tr>
<tr>
<td>bAutoPadding</td>
<td>BOOL</td>
<td>If true, the input image is automatically padded (with 0s) to optimal size if required, to speed up dft (Creates a temporary copy so that ipSpatialImage content stays unchanged, which also requires some additional computation power. Therefore, it is recommended to compare execution times with and without padding.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

**HRESULT [135]**

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.11.10 **F_VN_InverseDft**

Compute the inverse DFT for a given frequency image.

#### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_InverseDft : HRESULT
VAR_INPUT
    ipFrequencyImage : ITCvNImage;
    ipSpatialImage : Reference To ITCvNImage;
    bRealOutput : BOOL;
    hrPrev : HRESULT;
END_VAR
```

**F_VN_InverseDft**

- `ipFrequencyImage` (ITCvNImage): Frequency image to be transformed.
- `ipSpatialImage` (Reference To ITCvNImage): Original spatial image.
- `bRealOutput` (BOOL): Boolean indicating whether the output should be real or complex.
- `hrPrev` (HRESULT): HRESULT indicating the result of previous operations.

The function computes the inverse discrete Fourier transform (IDFT) of the frequency image `ipFrequencyImage` and stores the result in the spatial image `ipSpatialImage`. If `bRealOutput` is false, the result will be a complex image; if true, it will be a real image. The `hrPrev` parameter is an HRESULT indicating the result of previous operations.
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipFrequencyImage</td>
<td>ITcVnImage [383]</td>
<td>Source frequency image (elements of type REAL or LREAL, 1 (packed CCS) or 2 (Re + Im) channels)</td>
</tr>
<tr>
<td>ipSpatialImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (Same type as ipFrequencyImage, but number of channels can vary depending on bRealOutput. An appropriate image will be created if required.)</td>
</tr>
<tr>
<td>bRealOutput</td>
<td>BOOL</td>
<td>Only relevant if ipFrequencyImage has 2 channels. If true, the result image will have only 1 channel. Otherwise, the result image will have 2 separate channels (Re + Im). If ipFrequencyImage has only 1 channel, packed CCS format is assumed and the result image will always have only 1 channel.</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.11.11 F_VN_OptimalDftSize

Computes the optimal image size for applying a dft (can lead to better performance).

Syntax

Definition:

FUNCTION F_VN_OptimalDftSize : HRESULT
VAR_INPUT
  ipImage : ITcVnImage;
  nOptWidth : Reference To UDINT;
  nOptHeight : Reference To UDINT;
  hrPrev : HRESULT;
END_VAR
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage</td>
<td>ITcVnImage [383]</td>
<td>Source image for which to compute the optimal width and height</td>
</tr>
<tr>
<td>nOptWidth</td>
<td>Reference To UDINT</td>
<td>Returns the optimal width of the image</td>
</tr>
<tr>
<td>nOptHeight</td>
<td>Reference To UDINT</td>
<td>Returns the optimal height of the image</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCESSED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.11.12 F_VN_PadImageBorder

```plaintext
F_VN_PadImageBorder
```

```plaintext
ipSrcImage    ITcVnImage
ipPaddedImage Reference To ITcVnImage
nTopBorder    UDINT
nBottomBorder UDINT
nLeftBorder   UDINT
nRightBorder  UDINT
hrPrev        HRESULT
```

Add padding (zeros) around the original image borders.

### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_PadImageBorder : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipPaddedImage : Reference To ITcVnImage;
    nTopBorder : UDINT;
    nBottomBorder : UDINT;
    nLeftBorder : UDINT;
    nRightBorder : UDINT;
    hrPrev : HRESULT;
END_VAR
```
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image</td>
</tr>
<tr>
<td>ipPaddedImage</td>
<td>Reference To</td>
<td>Padded destination image (Same type as ipSrcImage, an appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>nTopBorder</td>
<td>UDINT</td>
<td>Padding height in pixels above top border</td>
</tr>
<tr>
<td>nBottomBorder</td>
<td>UDINT</td>
<td>Padding height in pixels below bottom border</td>
</tr>
<tr>
<td>nLeftBorder</td>
<td>UDINT</td>
<td>Padding width in pixels before left border</td>
</tr>
<tr>
<td>nRightBorder</td>
<td>UDINT</td>
<td>Padding width in pixels after right border</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.11.13 F_VN_PadImageBorderExp

Add padding around the original image borders. (expert function)

Syntax

Definition:

FUNCTION F_VN_PadImageBorderExp : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipPaddedImage : Reference To ITcVnImage;
    nTopBorder : UDINT;
    nBottomBorder : UDINT;
    nLeftBorder : UDINT;
    nRightBorder : UDINT;
    ePaddingType : ETCvNBorderInterpolationMethod;
    aPaddingValue : Reference To TcVnVector4_LREAL;
    hrPrev : HRESULT;
END_VAR
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipPaddedImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Padded destination image (Same type as ipSrcImage, an appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>nTopBorder</td>
<td>UDINT</td>
<td>Padding height in pixels above top border</td>
</tr>
<tr>
<td>nBottomBorder</td>
<td>UDINT</td>
<td>Padding height in pixels below bottom border</td>
</tr>
<tr>
<td>nLeftBorder</td>
<td>UDINT</td>
<td>Padding width in pixels before left border</td>
</tr>
<tr>
<td>nRightBorder</td>
<td>UDINT</td>
<td>Padding width in pixels after right border</td>
</tr>
<tr>
<td>ePaddingType</td>
<td>ETcVnBorderInterpolation Method [157]</td>
<td>Specifies how the pixel values of the padding area are determined</td>
</tr>
<tr>
<td>aPaddingValue</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Specifies the padding value if CONSTANT is used</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.12 Geometric and Coordinate Transformations

This group contains functions for transforming images and pixels.

#### Functions

**Calibration [943]**

- Intrinsic and extrinsic calibration
- Compensation of optical distortions
- Transformation between image and world coordinates

**Affine Transformation**

- F_VN_ApplyRotationToAffineTransformation(Exp) [977]
- F_VN_ApplyScalingToAffineTransformation(Exp) [978]
- F_VN_ApplyTranslationToAffineTransformation(Exp) [980]
- F_VN_ApplyYAxisInversionToAffineTransformation(Exp) [982]
- F_VN_GenerateAffineTransformationUnitMatrix2D [1001]
- F_VN_GetAffineTransformation [1002]
• F_VN_GetAffineTransformation2D(Exp) [1003]
• F_VN_InvertAffineTransformation [1008]
• F_VN_WarpAffine(Exp) [1017]
• F_VN_WarpAffine_Container [1018]
• F_VN_WarpAffine_Point [1019]

Transformation between cartesian and polar coordinates

• F_VN_ConvertCartesianToPolarAngleImage(Exp) [984]
• F_VN_ConvertCartesianToPolarAngles(Exp) [985]
• F_VN_ConvertCartesianToPolarImages(Exp) [987]
• F_VN_ConvertCartesianToPolarMagnitudeImage [989]
• F_VN_ConvertCartesianToPolarMagnitudes [990]
• F_VN_ConvertCartesianToPolarPoints(Exp) [991]
• F_VN_ConvertPolarToCartesianImages(Exp) [993]
• F_VN_ConvertPolarToCartesianPoints(Exp) [995]
• F_VN_RemapImageToLogPolarSpace(Exp) [1012]
• F_VN_RemapImageToPolarSpace(Exp) [1014]

Simple image operations

• F_VN_FlipImage [1001]
• F_VN_PyramidDown [1010]
• F_VN_PyramidUp [1011]
• F_VN_ResizeImage [1015]
• F_VN_RotateImage [1016]

Perspective transformation

• F_VN_DecomposeHomography(Exp) [997]
• F_VN_GetPerspectiveTransformation [1005]
• F_VN_Homography(Exp) [1006]
• F_VN_PerspectiveTransformation [1009]
• F_VN_WarpPerspective(Exp) [1021]
• F_VN_WarpPerspective_Container [1022]
• F_VN_WarpPerspective_Point [1023]
• F_VN_WarpPerspective_Rectangle [1024]

6.4.12.1 Calibration

This subgroup contains functions for the correlation between pixels and real world pixels. They are usually used together with the measurement functions [1215].

Functions

Calibration

• F_VN_CalibrateCamera(Exp) [944]
• F_VN_CalibrateCameraPlanar(Exp) [953]
• F_VN_CalibrateCameraManually \[952\]
• F_VN_CalibrateLinescanCamera(Exp) \[956\]
• F_VN_DetectPatternPoints(Exp) \[966\]
• F_VN_SortDetectedPatternPoints \[970\]

Compensation of lens distortion
• F_VN_CompensateLensDistortion(Exp) \[960\]
• F_VN_CompensateLensDistortionForPoints(Exp) \[963\]

Transformation between image and world coordinates
• F_VN_TransformCoordinatesImageToWorld \[972\]
• F_VN_TransformCoordinatesWorldToImage \[976\]
• F_VN_TransformCoordinatesPlanar \[974\]
• F_VN_ImagPointsWorldDistance \[976\]

6.4.12.1.1 \textbf{F_VN_CalibrateCamera}

\begin{tikzpicture}
  \node[text width=0.9\textwidth,align=left] {
    F_VN_CalibrateCamera \hspace{20pt} HRESULT F_VN_CalibrateCamera
  
  ipSrcImage \hspace{5pt} ITcVnImage
  ipReferencePoints \hspace{5pt} ITcVnContainer
  aCameraMatrix \hspace{5pt} Reference To TcVnMatrix3x3_LREAL
  aDistortionCoefficients \hspace{5pt} Reference To TcVnArray8_LREAL
  aRotationMatrix \hspace{5pt} Reference To TcVnMatrix3x3_LREAL
  aTranslationVector \hspace{5pt} Reference To TcVnVector3_LREAL
  fReprojError \hspace{5pt} Reference To LREAL
  stBlobDetectionParams \hspace{5pt} Reference To TcVnParamsBlobDetection
  stCalibrationOptions \hspace{5pt} Reference To TcVnCameraCalibrationOptions
  hrPrev \hspace{5pt} HRESULT

  Compute the camera parameters (intrinsic + extrinsic) by evaluating an image containing a calibration pattern (circles).
  Can use available TwinCAT Job Tasks for executing parallel code regions.

  Syntax

  Definition:

  FUNCTION F_VN_CalibrateCamera : HRESULT
  VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipReferencePoints : ITcVnContainer;
    aCameraMatrix : Reference To TcVnMatrix3x3_LREAL;
    aDistortionCoefficients : Reference To TcVnArray8_LREAL;
    aRotationMatrix : Reference To TcVnMatrix3x3_LREAL;
    aTranslationVector : Reference To TcVnVector3_LREAL;
    fReprojError : Reference To LREAL;
    stBlobDetectionParams : Reference To TcVnParamsBlobDetection;
    stCalibrationOptions : Reference To TcVnCameraCalibrationOptions;
    hrPrev : HRESULT;
  END_VAR
\end{tikzpicture}
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image containing a calibration pattern (1 channel, USINT elements)</td>
</tr>
<tr>
<td>ipReferencePoints</td>
<td>ITcVnContainer [345]</td>
<td>Reference calibration pattern point positions (ContainerType_Vector_TcVnPoint3_REAL)</td>
</tr>
<tr>
<td>aCameraMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>Returns the camera matrix</td>
</tr>
<tr>
<td>aDistortionCoefficients</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>Returns the lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]</td>
</tr>
<tr>
<td>aRotationMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>Returns the rotation matrix</td>
</tr>
<tr>
<td>aTranslationVector</td>
<td>Reference To TcVnVector3_LREAL [153]</td>
<td>Returns the translation vector</td>
</tr>
<tr>
<td>fReprojError</td>
<td>Reference To LREAL</td>
<td>Returns the reprojection error</td>
</tr>
<tr>
<td>stBlobDetectionParams</td>
<td>Reference To TcVnParamsBlobDetection [211]</td>
<td>Parameters for the internally used F_VN_DetectBlobs function</td>
</tr>
<tr>
<td>stCalibrationOptions</td>
<td>Reference To TcVnCameraCalibrationOptions [205]</td>
<td>Calibration options</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Further information

The function F_VN_CalibrateCamera calculates the intrinsic and extrinsic camera parameters on the basis of an input image containing a calibration pattern. The elements of the calibration pattern must be detectable as a blob.

### Input image

The input image ipSrcImage must be a 1-channel grayscale image with a bit depth of 8 bits. This image must contain a calibration pattern that matches the reference points ipReferencePoints.

The expert version F_VN_CalibrateCameraExp [947] is required for a calibration with several input images.

### Reference points

The container ipReferencePoints must contain the reference points that are illustrated on the input image ipSrcImage as the calibration pattern. The points must be specified as 3-dimensional points of the type REAL so that the container is of the type ContainerType_Vector_TcVnPoint3_REAL.

The reference points can be generated as follows:

- Manual creation of an XML file and reading with the function block FB_VN_ReadCalibrationPattern [1301].
• Configuration in the calibration assistants [103], saving in the TcCOM object by clicking Write Results and reading with the method GetCalibPatternRef [133] of the function block FB VN GevCameraControl [132]. The set center point is taken into account.
• Inserting reference points individually into a container.

Matrices of the camera parameters

These matrices return the calculated parameters of the camera model. They can then be used for the compensation of the objective distortions and for the conversion between pixel coordinates and world coordinates.

- Camera matrix
- Distortion coefficients
- Rotation matrix
- Translation vector

**Rounding errors**

On account of rounding errors, the calculated parameter values can deviate slightly from the values calculated in the calibration assistants.

**Reprojection errors**

**Parameters for blob detection**

- No chessboard pattern

As the reference points are detected as a blob structure, the use of chessboard patterns is not possible.

**Options for calibration**

- Default settings

To get the default settings from the calibration wizard [103], the bits bFixAspectRatio, bFixPrincipalPoint, bFixK5, bFixK6 and bRationalModel must be set to TRUE.

bFixPrincipalPoint = "Fix Center Point"

The bit bFixPrincipalPoint corresponds to the option "Fix Center Point" in the camera calibration assistant.

**Samples**

- Calibration Assistant [144]

**Required License**

TC3 Vision Metrology 2D

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.12.1.2 \texttt{F\_VN\_CalibrateCameraExp}

\begin{verbatim}
FUNCTION F_VN_CalibrateCameraExp : HRESULT
VAR_INPUT
  pSrcImages : PVOID;
  nSrcArraySize : UINT;
  ipReferencePoints : ITCvNContainer;
  aCameraMatrix : Reference To TcVnMatrix3x3_LREAL;
  aDistortionCoefficients : Reference To TcVnArray8_LREAL;
  aRotationMatrix : Reference To TcVnMatrix3x3_LREAL;
  aTranslationVector : Reference To TcVnVector3_LREAL;
  fReprojError : Reference To LREAL;
  stBlobDetectionParams : Reference To TcVnParamsBlobDetection;
  stCalibrationOptions : Reference To TcVnCameraCalibrationOptions;
  hrPrev : HRESULT;
END_VAR
\end{verbatim}

Compute the camera parameters (intrinsic + extrinsic) by evaluating images containing a calibration pattern (circles). The extrinsic parameters are computed for the first image in the array. (expert function)

Can use available TwinCAT Job Tasks for executing parallel code regions.

\textbf{Syntax}

\textbf{Definition:}

\begin{verbatim}
FUNCTION F_VN_CalibrateCameraExp : HRESULT
VAR_INPUT
  pSrcImages : PVOID;
  nSrcArraySize : UINT;
  ipReferencePoints : ITCvNContainer;
  aCameraMatrix : Reference To TcVnMatrix3x3_LREAL;
  aDistortionCoefficients : Reference To TcVnArray8_LREAL;
  aRotationMatrix : Reference To TcVnMatrix3x3_LREAL;
  aTranslationVector : Reference To TcVnVector3_LREAL;
  fReprojError : Reference To LREAL;
  stBlobDetectionParams : Reference To TcVnParamsBlobDetection;
  stCalibrationOptions : Reference To TcVnCameraCalibrationOptions;
  hrPrev : HRESULT;
END_VAR
\end{verbatim}
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pSrcImages</td>
<td>PVOID</td>
<td>Pointer to an array of source images, each containing the same calibration pattern (1 channel, USINT elements)</td>
</tr>
<tr>
<td>nSrcArraySize</td>
<td>UINT</td>
<td>pSrcImages array size</td>
</tr>
<tr>
<td>ipReferencePoints</td>
<td>ITcVnContainer</td>
<td>Reference calibration pattern point positions (ContainerType_Vector_TcVnPoint3_REAL)</td>
</tr>
<tr>
<td>aCameraMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>Returns the camera matrix</td>
</tr>
<tr>
<td>aDistortionCoefficients</td>
<td>Reference To TcVnArray8_LREAL [153]</td>
<td>Returns the lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]</td>
</tr>
<tr>
<td>aRotationMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>Returns the rotation matrix</td>
</tr>
<tr>
<td>aTranslationVector</td>
<td>Reference To TcVnVector3_LREAL [153]</td>
<td>Returns the translation vector</td>
</tr>
<tr>
<td>fReprojError</td>
<td>Reference To LREAL</td>
<td>Returns the reprojection error</td>
</tr>
<tr>
<td>stBlobDetectionParams</td>
<td>Reference To TcVnParamsBlobDetection [211]</td>
<td>Parameters for the internally used F_VN_DetectBlobs function</td>
</tr>
<tr>
<td>stCalibrationOptions</td>
<td>Reference To TcVnCameraCalibrationOptions [205]</td>
<td>Calibration options</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Further information

The function `F_VN_CalibrateCameraExp` is the expert version of `F_VN_CalibrateCamera [944]`. It enables the use of several input images for the calibration of the camera.

## Parameter

### Input images

The input images for the camera calibration are transferred to the function with the help of the pointer `pSrcImages` and the number of images `nSrcArraySize`.

### Pointer handling

Make sure that the specification of the pointer and the array size match the actual data. Otherwise an impermissible memory access may occur, resulting in a system crash.
Reference points

The container `ipReferencePoints` must contain the reference points that are illustrated on the input image `ipSrcImage` as the calibration pattern. The points must be specified as 3-dimensional points of the type `REAL` so that the container is of the type `ContainerType_Vector_TcVnPoint3_REAL`.

The reference points can be generated as follows:

- Manual creation of an XML file and reading with the function block `FB_VN_ReadCalibrationPattern`.
- Configuration in the calibration assistants, saving in the TcCOM object by clicking `Write Results` and reading with the method `GetCalibPatternRef` of the function block `FB_VN_GevCameraControl`. The set center point is taken into account.
- Inserting reference points individually into a container.

Matrices of the camera parameters

These matrices return the calculated parameters of the camera model. They can then be used for the compensation of the objective distortions and for the conversion between pixel coordinates and world coordinates.

- Camera matrix
- Distortion coefficients
- Rotation matrix
- Translation vector

Rounding errors

On account of rounding errors, the calculated parameter values can deviate slightly from the values calculated in the calibration assistants.

Reprojection errors

Parameters for blob detection

- No chessboard pattern

As the reference points are detected as a blob structure, the use of chessboard patterns is not possible.

Options for calibration

- Default settings

To get the default settings from the calibration wizard, the bits `bFixAspectRatio`, `bFixPrincipalPoint`, `bFixK5`, `bFixK6` and `bRationalModel` must be set to TRUE.

`bFixPrincipalPoint = „Fix Center Point“`

The bit `bFixPrincipalPoint` corresponds to the option "Fix Center Point" in the camera calibration assistant.

Application

Samples

- Calibration Assistant

Required License

TC3 Vision Metrology 2D
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.12.1.3 F_VN_CalibrateCameraExp2

**F_VN_CalibrateCameraExp2**

```
FUNCTION F_VN_CalibrateCameraExp2 : HRESULT

Syntax

Can use available TwinCAT Job Tasks for executing parallel code regions.

Compute the camera parameters (intrinsic + extrinsic) by evaluating images containing a calibration pattern (circles). The extrinsic parameters are computed for the first image in the array. (expert function)

Can use available TwinCAT Job Tasks for executing parallel code regions.

**Definition:**

```
FUNCTION F_VN_CalibrateCameraExp2 : HRESULT
VAR_INPUT
    pSrcImages    : PVOID;
    nSrcArraySize : UINT;
    ipReferencePoints : ITCvnContainer;
    aCameraMatrix : Reference To TcVnMatrix3x3_LREAL;
    aDistortionCoefficients : Reference To TcVnMatrix8x8_LREAL;
    aRotationMatrix : Reference To TcVnMatrix3x3_LREAL;
    aTranslationVector : Reference To TcVnVector3_LREAL;
    fReprojError : Reference To LREAL;
    stBlobDetectionParams : Reference To TcVnParamsBlobDetection;
    stCalibrationOptions : Reference To TcVnCameraCalibrationOptions;
    bSubpixelAccuracy : BOOL;
    eEdgeDirection : ETCvnEdgeDirection;
    fMinStrength : REAL;
    nMaxThickness : UDINT;
    nSubpixelsIterations : UDINT;
    nSearchLines : UDINT;
    fApproxPrecision : REAL;
    eAlgorithm : ETCvnEdgeDetectionAlgorithm;
    hrPrev : HRESULT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pSrcImages</td>
<td>PVOID</td>
<td>Pointer to an array of source images, each containing the same calibration pattern (1 channel, USINT elements)</td>
</tr>
<tr>
<td>nSrcArraySize</td>
<td>UINT</td>
<td>pSrcImages array size</td>
</tr>
<tr>
<td>ipReferencePoints</td>
<td>ITcVnContainer [345]</td>
<td>Reference calibration pattern point positions (ContainerType_Vector_TcVnPoint3_REAL)</td>
</tr>
<tr>
<td>aCameraMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>Returns the camera matrix</td>
</tr>
<tr>
<td>aDistortionCoefficients</td>
<td>Reference To TcVnArray8_LREAL [153]</td>
<td>Returns the lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]</td>
</tr>
<tr>
<td>aRotationMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>Returns the rotation matrix</td>
</tr>
<tr>
<td>aTranslationVector</td>
<td>Reference To TcVnVector3_LREAL [153]</td>
<td>Returns the translation vector</td>
</tr>
<tr>
<td>fReprojError</td>
<td>Reference To LREAL</td>
<td>Returns the reprojection error</td>
</tr>
<tr>
<td>stBlobDetectionParams</td>
<td>Reference To TcVnParamsBlobDetection [211]</td>
<td>Parameters for the internally used F_VN_DetectBlobs function</td>
</tr>
<tr>
<td>stCalibrationOptions</td>
<td>Reference To TcVnCameraCalibrationOptions [205]</td>
<td>Calibration options</td>
</tr>
<tr>
<td>bSubpixelAccuracy</td>
<td>BOOL</td>
<td>If true, the pattern points are detected with subpixel accuracy</td>
</tr>
<tr>
<td>eEdgeDirection</td>
<td>ETcVnEdgeDirection [186]</td>
<td>Specification of the edge direction to search for (from center to outside ellipse)</td>
</tr>
<tr>
<td>fMinStrength</td>
<td>REAL</td>
<td>Specification of the minimum strength (intensity difference) of the edge to search for</td>
</tr>
<tr>
<td>nMaxThickness</td>
<td>UDINT</td>
<td>Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels</td>
</tr>
<tr>
<td>nSubpixelsIterations</td>
<td>UDINT</td>
<td>Specifies the number of subpixels (for INTERPOLATION, 10 - 20 usually is sufficient) or maximum number of iterations for optimizing the parameters (for APPROX_ERF and APPROX_GAUSSIAN, 50 - 100 usually is sufficient)</td>
</tr>
<tr>
<td>nSearchLines</td>
<td>UDINT</td>
<td>Specifies the amount of search lines, which are equally distributed in all directions (must be &gt;= 8 and a multiple of 4)</td>
</tr>
<tr>
<td>fApproxPrecision</td>
<td>REAL</td>
<td>Specifies the approximation precision for APPROX_ERF and APPROX_GAUSSIAN (0.001 usually is sufficient, unused for INTERPOLATION)</td>
</tr>
<tr>
<td>eAlgorithm</td>
<td>ETcVnEdgeDetectionAlgorithm [185]</td>
<td>Selection of the edge detection algorithm</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]
**Required License**
TC3 Vision Metrology 2D

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

---

### 6.4.12.1.4 F_VN_CalibrateCameraManually

Compute the camera parameters (intrinsic + extrinsic). The extrinsic parameters are computed for the first inner container.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_CalibrateCameraManually : HRESULT
VAR_INPUT
  ipImagePoints : ITCvnContainer;
  nImageWidth : UDINT;
  nImageHeight : UDINT;
  ipReferencePoints : ITCvnContainer;
  aCameraMatrix : Reference To TcVnMatrix3x3_LREAL;
  aDistortionCoefficients : Reference To TcVnArray8_LREAL;
  aRotationMatrix : Reference To TcVnMatrix3x3_LREAL;
  aTranslationVector : Reference To TcVnVector3_LREAL;
  fReprojError : Reference To LREAL;
  stCalibrationOptions : Reference To TcVnCameraCalibrationOptions;
  hrPrev : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImagePoints</td>
<td>ITcVnContainer [345]</td>
<td>Provided image points of multiple images (ContainerType_Vector_Vector_TcVnPoint2_REAL, each inner container with at least 6 points). The container and point order must match ipReferencePoints.</td>
</tr>
<tr>
<td>nImageWidth</td>
<td>UDINT</td>
<td>Image width</td>
</tr>
<tr>
<td>nImageHeight</td>
<td>UDINT</td>
<td>Image height</td>
</tr>
<tr>
<td>ipReferencePoints</td>
<td>ITcVnContainer [345]</td>
<td>Reference world points (ContainerType_Vector_Vector_TcVnPoint3_REAL). The number of inner containers and their amount of points must match ipImagePoints.</td>
</tr>
<tr>
<td>aCameraMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>Returns the camera matrix</td>
</tr>
<tr>
<td>aDistortionCoefficients</td>
<td>Reference To TcVnArray8_LREAL [153]</td>
<td>Returns the lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]</td>
</tr>
<tr>
<td>aRotationMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>Returns the rotation matrix (for the first inner container)</td>
</tr>
<tr>
<td>aTranslationVector</td>
<td>Reference To TcVnVector3_LREAL [153]</td>
<td>Returns the translation vector (for the first inner container)</td>
</tr>
<tr>
<td>fReprojError</td>
<td>Reference To LREAL</td>
<td>Returns the reprojection error</td>
</tr>
<tr>
<td>stCalibrationOptions</td>
<td>Reference To TcVnCameraCalibrationOptions [205]</td>
<td>Calibration options</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Metrology 2D

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.12.1.5 F_VN_CalibrateCameraPlanar

```c
F_VN_CalibrateCameraPlanar(ipSrcImage ITcVnImage, aMarkerPositions Reference To TcVnArray4_Point2_REAL, aTransformationMatrix Reference To TcVnMatrix3x3_LREAL, hrPrev HRESULT) HRESULT F_VN_CalibrateCameraPlanar
```
Calibrate camera using a planar calibration pattern comprised of four circles marking the corners of a rectangle. Can use available TwinCAT Job Tasks for executing parallel code regions.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_CalibrateCameraPlanar : HRESULT
VAR_INPUT
   ipSrcImage : ITcVnImage;
   aMarkerPositions : Reference To TcVnArray4_Point2_REAL;
   aTransformationMatrix : Reference To TcVnMatrix3x3_LREAL;
   hrPrev : HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Binary source image (background zero, markers non-zero)</td>
</tr>
<tr>
<td>aMarkerPositions</td>
<td>Reference To TcVnArray4_Point2_REAL [153]</td>
<td>Marker positions within their plane using world units (right-handed coordinate system)</td>
</tr>
<tr>
<td>aTransformationMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>Returns a 3-by-3 matrix for the transformation from image coordinates into the reference coordinate system</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

**HRESULT [135]**

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.12.1.6 F_VN_CalibrateCameraPlanarExp**

```c
F_VN_CalibrateCameraPlanarExp
   ipSrcImage   ITCvNImage
   aMarkerPositions Reference To TcVnArray4_Point2_REAL
   aTransformationMatrix Reference To TcVnMatrix3x3_LREAL
   aMarkerImagePositions Reference To TcVnArray4_Point2_REAL
   nMinMarkerArea    UDINT
   nMaxMarkerArea    UDINT
   fMaxMarkerEccentricity LREAL
   hrPrev            HRESULT
```

Calibrate camera using a planar calibration pattern comprised of four circles marking the corners of a rectangle. (expert function) Can use available TwinCAT Job Tasks for executing parallel code regions.
Syntax

Definition:

FUNCTION F_VN_CalibrateCameraPlanarExp : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  aMarkerPositions : Reference To TcVnArray4_Point2_REAL;
  aTransformationMatrix : Reference To TcVnMatrix3x3_LREAL;
  aMarkerImagePositions : Reference To TcVnArray4_Point2_REAL;
  nMinMarkerArea : UDINT;
  nMaxMarkerArea : UDINT;
  fMaxMarkerEccentricity : LREAL;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Binary source image (background zero, markers non-zero)</td>
</tr>
<tr>
<td>aMarkerPositions</td>
<td>Reference To TcVnArray4_Point2_REAL</td>
<td>Marker positions within their plane using world units (right-handed coordinate system)</td>
</tr>
<tr>
<td>aTransformationMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL</td>
<td>Returns a 3-by-3 matrix for the transformation from image coordinates into the reference coordinate system</td>
</tr>
<tr>
<td>aMarkerImagePositions</td>
<td>Reference To TcVnArray4_Point2_REAL</td>
<td>Returns the marker positions in image coordinates (left-handed coordinate system)</td>
</tr>
<tr>
<td>nMinMarkerArea</td>
<td>UDINT</td>
<td>Minimum area for the detection of markers (Smaller markers are ignored.)</td>
</tr>
<tr>
<td>nMaxMarkerArea</td>
<td>UDINT</td>
<td>Maximum area for the detection of markers (Larger markers are ignored.)</td>
</tr>
<tr>
<td>fMaxMarkerEccentricity</td>
<td>LREAL</td>
<td>Maximum eccentricity for the detection of markers (The eccentricity measures deviations from a round shape (eccentricity = 0: round object; eccentricity = 1.0: linear object).)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
### 6.4.12.1.7 F_VN_CalibrateLinescanCamera

Compute the camera parameters (intrinsic + extrinsic) by evaluating images containing a line calibration pattern, consisting of alternating vertical and diagonal lines, each diagonal line having the same constant slope: \( |/|...| \). To be compatible to 2d calibration results, the y-position of the image line is set to half sensor width, so that it would be the center line of a virtual 2d sensor with equal width and height. The extrinsic parameters are computed for the first image in the array.

Can use available TwinCAT Job Tasks for executing parallel code regions.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_CalibrateLinescanCamera : HRESULT
VAR_INPUT
    pSrcImages : PVOID;
    nSrcArraySize : UINT;
    ipPatternLineOrigins : ITcVnContainer;
    fSlopeRad : LREAL;
    aCameraMatrix : Reference To TcVnMatrix3x3_LREAL;
    aDistortionCoefficients : Reference To TcVnArray8_LREAL;
    aRotationMatrix : Reference To TcVnMatrix3x3_LREAL;
    aTranslationVector : Reference To TcVnVector3_LREAL;
    fReprojError : Reference To LREAL;
    ipPatternLineOrigins : ITcVnContainer;
    nSrcArraySize : UINT;
    pSrcImages : PVOID;
    VAR_INPUT
    hrPrev : HRESULT
    F_VN_CalibrateLinescanCamera
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pSrcImages</td>
<td>PVOID</td>
<td>Pointer to an array of 1 or more source images (1 channel). If the images have more than 1 row, each row must contain an image of the same pattern position.</td>
</tr>
<tr>
<td>nSrcArraySize</td>
<td>UINT</td>
<td>pSrcImages array size</td>
</tr>
<tr>
<td>ipPatternLineOrigins</td>
<td>ITcVnContainer</td>
<td>X position of the pattern line origins (ContainerType_Vector_REAL, usually same origin for vertical and following diagonal line)</td>
</tr>
<tr>
<td>fSlopeRad</td>
<td>LREAL</td>
<td>Slope of the diagonal line in rad (&gt; 0, &lt; PI/2), relative to the vertical line (i.e., 0 would be a vertical line, PI/2 a horizontal line)</td>
</tr>
<tr>
<td>aCameraMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL</td>
<td>Returns the camera matrix</td>
</tr>
<tr>
<td>aDistortionCoefficients</td>
<td>Reference To TcVnArray8_LREAL</td>
<td>Returns the lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]</td>
</tr>
<tr>
<td>aRotationMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL</td>
<td>Returns the rotation matrix</td>
</tr>
<tr>
<td>aTranslationVector</td>
<td>Reference To TcVnVector3_LREAL</td>
<td>Returns the translation vector</td>
</tr>
<tr>
<td>fReprojError</td>
<td>Reference To LREAL</td>
<td>Returns the reprojection error</td>
</tr>
<tr>
<td>stCalibrationOptions</td>
<td>Reference To TcVnCameraCalibrationOptions</td>
<td>Calibration options (recommended to set bFixAspectRatio, bFixPrincipalPoint, bZeroTangentDist)</td>
</tr>
<tr>
<td>eEdgeDirection</td>
<td>ETcVnEdgeDirection</td>
<td>Specification of the edge direction to search for</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Metrology 2D

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
### 6.4.12.1.8 F_VN_CalibrateLinescanCameraExp

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pSrcImages</td>
<td>PVOID</td>
<td>Source images</td>
</tr>
<tr>
<td>nSrcArraySize</td>
<td>UINT</td>
<td>Source array size</td>
</tr>
<tr>
<td>ipPatternLineOrigins</td>
<td>ITcVnContainer</td>
<td>Image pattern line origins</td>
</tr>
<tr>
<td>fSlopeRad</td>
<td>LREAL</td>
<td>Slope radius</td>
</tr>
<tr>
<td>aCameraMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL</td>
<td>Camera matrix</td>
</tr>
<tr>
<td>aDistortionCoefficients</td>
<td>Reference To TcVnArray8_LREAL</td>
<td>Distortion coefficients</td>
</tr>
<tr>
<td>aRotationMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL</td>
<td>Rotation matrix</td>
</tr>
<tr>
<td>aTranslationVector</td>
<td>Reference To TcVnVector3_LREAL</td>
<td>Translation vector</td>
</tr>
<tr>
<td>fReprojError</td>
<td>Reference To LREAL</td>
<td>Reprojection error</td>
</tr>
<tr>
<td>stCalibrationOptions</td>
<td>Reference To TcVnCameraCalibrationOptions</td>
<td>Calibration options</td>
</tr>
<tr>
<td>eEdgeDirection</td>
<td>EtcVnEdgeDirection</td>
<td>Edge direction</td>
</tr>
<tr>
<td>fMinStrength</td>
<td>REAL</td>
<td>Minimum strength</td>
</tr>
<tr>
<td>nMaxSearchLines</td>
<td>UDINT</td>
<td>Maximum search lines</td>
</tr>
<tr>
<td>nMaxThickness</td>
<td>UDINT</td>
<td>Maximum thickness</td>
</tr>
<tr>
<td>nSubpixelIterations</td>
<td>UDINT</td>
<td>Subpixel iterations</td>
</tr>
<tr>
<td>fApproxPrecision</td>
<td>REAL</td>
<td>Approximation precision</td>
</tr>
<tr>
<td>eAlgorithm</td>
<td>EtcVnEdgeDetectionAlgorithm</td>
<td>Algorithm</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>Previous result</td>
</tr>
</tbody>
</table>

Compute the camera parameters (intrinsic + extrinsic) by evaluating images containing a line calibration pattern, consisting of alternating vertical and diagonal lines, each diagonal line having the same constant slope: \([\L_1, \L_2, \L_3, \ldots]\). To be compatible to 2d calibration results, the y-position of the image line is set to half sensor width, so that it would be the center line of a virtual 2d sensor with equal width and height. The extrinsic parameters are computed for the first image in the array. (expert function)

Can use available TwinCAT Job Tasks for executing parallel code regions.

**Syntax**

**Definition:**

```
FUNCTION F_VN_CalibrateLinescanCameraExp : HRESULT
VAR_INPUT
    pSrcImages : PVOID;
    nSrcArraySize : UINT;
    ipPatternLineOrigins : ITcVnContainer;
    fSlopeRad : LREAL;
    aCameraMatrix : Reference To TcVnMatrix3x3_LREAL;
    aDistortionCoefficients : Reference To TcVnArray8_LREAL;
    aRotationMatrix : Reference To TcVnMatrix3x3_LREAL;
    aTranslationVector : Reference To TcVnVector3_LREAL;
    fReprojError : Reference To LREAL;
    stCalibrationOptions : Reference To TcVnCameraCalibrationOptions;
    eEdgeDirection : EtcVnEdgeDirection;
    fMinStrength : REAL;
    nMaxSearchLines : UDINT;
    nMaxThickness : UDINT;
    nSubpixelIterations : UDINT;
    fApproxPrecision : REAL;
    eAlgorithm : EtcVnEdgeDetectionAlgorithm;
    hrPrev : HRESULT;
END_VAR
```
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pSrcImages</td>
<td>PVOID</td>
<td>Pointer to an array of 1 or more source images (1 channel). If the images have more than 1 row, each row must contain an image of the same pattern position.</td>
</tr>
<tr>
<td>nSrcArraySize</td>
<td>UINT</td>
<td>pSrcImages array size</td>
</tr>
<tr>
<td>ipPatternLineOrigins</td>
<td>ITcVnContainer</td>
<td>X position of the pattern line origins (ContainerType_Vector_REAL, usually same origin for vertical and following diagonal line)</td>
</tr>
<tr>
<td>fSlopeRad</td>
<td>LREAL</td>
<td>Slope of the diagonal line in rad (&gt; 0, &lt; PI/2), relative to the vertical line (i.e., 0 would be a vertical line, PI/2 a horizontal line)</td>
</tr>
<tr>
<td>aCameraMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL</td>
<td>Returns the camera matrix</td>
</tr>
<tr>
<td>aDistortionCoefficients</td>
<td>Reference To TcVnMatrix8_LREAL</td>
<td>Returns the lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]</td>
</tr>
<tr>
<td>aRotationMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL</td>
<td>Returns the rotation matrix</td>
</tr>
<tr>
<td>aTranslationVector</td>
<td>Reference To TcVnVector3_LREAL</td>
<td>Returns the translation vector</td>
</tr>
<tr>
<td>fReprojError</td>
<td>Reference To LREAL</td>
<td>Returns the reprojection error</td>
</tr>
<tr>
<td>stCalibrationOptions</td>
<td>Reference To TcVnCameraCalibrationOptions</td>
<td>Calibration options (recommended to set bFixAspectRatio, bFixPrincipalPoint, bZeroTangentDist)</td>
</tr>
<tr>
<td>eEdgeDirection</td>
<td>ETcVnEdgeDirection</td>
<td>Specification of the edge direction to search for</td>
</tr>
<tr>
<td>fMinStrength</td>
<td>REAL</td>
<td>Specification of the minimum strength (intensity difference) of the edge to search for</td>
</tr>
<tr>
<td>nMaxSearchLines</td>
<td>UDINT</td>
<td>Maximum number of search lines (equally distributed over the image height, at most 1 searchline per image row)</td>
</tr>
<tr>
<td>nMaxThickness</td>
<td>UDINT</td>
<td>Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels</td>
</tr>
<tr>
<td>nSubpixelsIterations</td>
<td>UDINT</td>
<td>Specifies the number of subpixels (for INTERPOLATION, 10 - 20 usually is sufficient) or maximum number of iterations for optimizing the parameters (for APPROX_ERF and APPROX_GAUSSIAN, 50 - 100 usually is sufficient)</td>
</tr>
<tr>
<td>fApproxPrecision</td>
<td>REAL</td>
<td>Specifies the approximation precision for APPROX_ERF and APPROX_GAUSSIAN (0.001 usually is sufficient, unused for INTERPOLATION)</td>
</tr>
<tr>
<td>eAlgorithm</td>
<td>ETcVnEdgeDetectionAlgorithm</td>
<td>Selection of the edge detection algorithm</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

 HRESULT [135]
6.4.12.1.9  F_VN_CompensateLensDistortion

Transforms an image to compensate the lens distortion. Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

FUNCTION F_VN_CompensateLensDistortion : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  aCameraMatrix : Reference To TcVnMatrix3x3_LREAL;
  aDistortionCoefficients : Reference To TcVnArray4_LREAL;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>aCameraMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>Camera matrix containing the intrinsic parameters</td>
</tr>
<tr>
<td>aDistortionCoefficients</td>
<td>Reference To TcVnArray4_LREAL [153]</td>
<td>Lens distortion coefficients [k1, k2, p1, p2]</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Metrology 2D
6.4.12.1.10 F_VN_CompensateLensDistortionExp1

Transforms an image to compensate the lens distortion. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

FUNCTION F_VN_CompensateLensDistortionExp1 : HRESULT
VAR_INPUT
  ipSrcImage : ITCvImage;
  ipDestImage : Reference To ITCvImage;
  aCameraMatrix : Reference To TcVnMatrix3x3_LREAL;
  aDistortionCoefficients : Reference To TcVnArray8_LREAL;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITCvImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>aCameraMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>Camera matrix containing the intrinsic parameters</td>
</tr>
<tr>
<td>aDistortionCoefficients</td>
<td>Reference To TcVnArray8_LREAL [153]</td>
<td>Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Metrology 2D

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.12.1.11  F_VN_CompenatesLensDistortionExp2

Transforms an image to compensate the lens distortion. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

FUNCTION F_VN_CompenatesLensDistortionExp2 : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    aCameraMatrix : Reference To TcVnMatrix3x3_LREAL;
    aDistortionCoefficients : Reference To TcVnArray8_LREAL;
    aNewCameraMatrix : Reference To TcVnMatrix3x3_LREAL;
    hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>aCameraMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>Camera matrix containing the intrinsic parameters</td>
</tr>
<tr>
<td>aDistortionCoefficients</td>
<td>Reference To TcVnArray8_LREAL [153]</td>
<td>Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]</td>
</tr>
<tr>
<td>aNewCameraMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>Allows additional scaling and shifting of the result image</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Metrology 2D

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
Transforms point coordinates to compensate the lens distortion.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_CompensateLensDistortionForPoints : HRESULT
VAR_INPUT
   ipSrcPoints : ITCvNContainer;
   ipDestPoints : Reference To ITCvNContainer;
   aCameraMatrix : Reference To TcVnMatrix3x3_LREAL;
   aDistortionCoefficients : Reference To TcVnArray4_LREAL;
   hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcPoints</td>
<td>ITCvNContainer [345]</td>
<td>Container with source point coordinates (ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_DINT)</td>
</tr>
<tr>
<td>ipDestPoints</td>
<td>Reference To ITCvNContainer</td>
<td>Returns a container with the transformed point coordinates (ContainerType_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>aCameraMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>Camera matrix containing the intrinsic parameters</td>
</tr>
<tr>
<td>aDistortionCoefficients</td>
<td>Reference To TcVnArray4_LREAL [153]</td>
<td>Lens distortion coefficients [k1, k2, p1, p2]</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Metrology 2D

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.12.1.13  F_VN_CompensateLensDistortionForPointsExp1

Transforms point coordinates to compensate the lens distortion. (expert function)

**Syntax**

**Definition:**

```c
FUNCTION F_VN_CompensateLensDistortionForPointsExp1 : HRESULT
VAR_INPUT
  ipSrcPoints : ITCvNContainer;
  ipDestPoints : Reference To ITCvNContainer;
  aCameraMatrix : Reference To TcVnMatrix3x3_LREAL;
  aDistortionCoefficients : Reference To TcVnArray8_LREAL;
  hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcPoints</td>
<td>ITCvNContainer</td>
<td>Container with source point coordinates (ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_DINT)</td>
</tr>
<tr>
<td>ipDestPoints</td>
<td>Reference To ITCvNContainer</td>
<td>Returns a container with the transformed point coordinates (ContainerType_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>aCameraMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL</td>
<td>Camera matrix containing the intrinsic parameters</td>
</tr>
<tr>
<td>aDistortionCoefficients</td>
<td>Reference To TcVnArray8_LREAL</td>
<td>Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Metrology 2D

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.12.1.14  F_VN_CompensateLensDistortionForPointsExp2

Transforms point coordinates to compensate the lens distortion. (expert function)

**Syntax**

Definition:

```c
FUNCTION F_VN_CompensateLensDistortionForPointsExp2 : HRESULT
VAR_INPUT
    ipSrcPoints : ITcVnContainer;
    ipDestPoints : Reference To ITcVnContainer;
    aCameraMatrix : Reference To TcVnMatrix3x3_LREAL;
    aDistortionCoefficients : Reference To TcVnArray8_LREAL;
    aNewCameraMatrix : Reference To TcVnMatrix3x3_LREAL;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcPoints</td>
<td>ITcVnContainer [345]</td>
<td>Container with source point coordinates (ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_DINT)</td>
</tr>
<tr>
<td>ipDestPoints</td>
<td>Reference To ITcVnContainer</td>
<td>Returns a container with the transformed point coordinates (ContainerType_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>aCameraMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>Camera matrix containing the intrinsic parameters</td>
</tr>
<tr>
<td>aDistortionCoefficients</td>
<td>Reference To TcVnArray8_LREAL [153]</td>
<td>Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]</td>
</tr>
<tr>
<td>aNewCameraMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>Allows additional shifting of the result point coordinates</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Metrology 2D

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.12.1.15  F_VN_DetectPatternPoints

Detects calibration pattern points (circles) within the provided image. Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```c
FUNCTION F_VN_DetectPatternPoints : HRESULT
VAR_INPUT
    ipSrcImage : ITCvImage;
    ipPatternPoints : Reference To ITCvContainer;
    stBlobDetectionParams : Reference To TcVnParamsBlobDetection;
    nNumberOfPoints : UDINT;
    hrPrev : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvImage</td>
<td>Source image (USINT, 1 channel)</td>
</tr>
<tr>
<td>ipPatternPoints</td>
<td>Reference To ITCvContainer</td>
<td>Returns the pattern point positions (ContainerType_Vector_TcVnPoint2_REAL; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>stBlobDetectionParams</td>
<td>Reference To TcVnParamsBlobDetection</td>
<td>Parameters to detect and filter contours.</td>
</tr>
<tr>
<td>nNumberOfPoints</td>
<td>UDINT</td>
<td>Expected number of pattern points</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

`HRESULT [135]`

Required License

TC3 Vision Metrology 2D

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.12.1.16 F_VN_DetectPatternPointsExp

F_VN_DetectPatternPointsExp

F_VN_DetectPatternPointsExp : HRESULT

VAR_INPUT

- ipSrcImage : ITcVnImage;
- ipPatternPoints : Reference To ITcVnContainer;
- stBlobDetectionParams : Reference To TcVnParamsBlobDetection;
- nNumberOfPoints : UDINT;
- fSizeDiffThresFactor : REAL;
- bSubpixelAccuracy : BOOL;
- eEdgeDirection : ETcVnEdgeDirection;
- fMinStrength : REAL;
- nMaxThickness : UDINT;
- nSubpixelsIterations : UDINT;
- nSearchLines : UDINT;
- fApproxPrecision : REAL;
- eAlgorithm : ETcVnEdgeDetectionAlgorithm;
- hrPrev : HRESULT;
END_VAR

Detections calibration pattern points (circles) within the provided image. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

FUNCTION F_VN_DetectPatternPointsExp : HRESULT

VAR_INPUT

- ipSrcImage : ITcVnImage;
- ipPatternPoints : Reference To ITcVnContainer;
- stBlobDetectionParams : Reference To TcVnParamsBlobDetection;
- nNumberOfPoints : UDINT;
- fSizeDiffThresFactor : REAL;
- bSubpixelAccuracy : BOOL;
- eEdgeDirection : ETcVnEdgeDirection;
- fMinStrength : REAL;
- nMaxThickness : UDINT;
- nSubpixelsIterations : UDINT;
- nSearchLines : UDINT;
- fApproxPrecision : REAL;
- eAlgorithm : ETcVnEdgeDetectionAlgorithm;
- hrPrev : HRESULT;
END_VAR
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvImage</td>
<td>Source image (USINT, 1 channel)</td>
</tr>
<tr>
<td>ipPatternPoints</td>
<td>Reference To ITCvImageContainer</td>
<td>Returns the pattern point positions (ContainerType_Vector_TcVnPoint2_REAL; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>stBlobDetectParams</td>
<td>Reference To TcVnParamsBlobDetection</td>
<td>Parameters to detect and filter contours.</td>
</tr>
<tr>
<td>nNumberOfPoints</td>
<td>UDINT</td>
<td>Expected number of pattern points</td>
</tr>
<tr>
<td>fSizeDiffThressFactor</td>
<td>REAL</td>
<td>Threshold for filtering contours by average size</td>
</tr>
<tr>
<td>bSubpixelAccuracy</td>
<td>BOOL</td>
<td>If true, the pattern points are detected with subpixel accuracy</td>
</tr>
<tr>
<td>eEdgeDirection</td>
<td>ETCvEdgeDirection</td>
<td>Specification of the edge direction to search for (from center to outside circle)</td>
</tr>
<tr>
<td>fMinStrength</td>
<td>REAL</td>
<td>Specification of the minimum strength (intensity difference) of the edge to search for</td>
</tr>
<tr>
<td>nMaxThickness</td>
<td>UDINT</td>
<td>Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels</td>
</tr>
<tr>
<td>nSubpixelsIterations</td>
<td>UDINT</td>
<td>Specifies the number of subpixels (for INTERPOLATION, 10 - 20 usually is sufficient) or maximum number of iterations for optimizing the parameters (for APPROX_ERF and APPROX_GAUSSIAN, 50 - 100 usually is sufficient)</td>
</tr>
<tr>
<td>nSearchLines</td>
<td>UDINT</td>
<td>Specifies the amount of search lines, which are equally distributed in all directions (must be &gt;= 8 and a multiple of 4)</td>
</tr>
<tr>
<td>fApproxPrecision</td>
<td>REAL</td>
<td>Specifies the approximation precision for APPROX_ERF and APPROX_GAUSSIAN (0.001 usually is sufficient, unused for INTERPOLATION)</td>
</tr>
<tr>
<td>eAlgorithm</td>
<td>ETCvEdgeDetectionAlgorithm</td>
<td>Selection of the edge detection algorithm</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Metrology 2D

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
### F_VN_ImagePointsWorldDistance

**Description:**
computes the distance in the world coordinate system between two points (in the same world coordinate z layer) given in image coordinates.

#### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_ImagePointsWorldDistance : HRESULT
VAR_INPUT
  aImagePoint1 : Reference To TcVnPoint2_REAL;
  aImagePoint2 : Reference To TcVnPoint2_REAL;
  fWorldDistance : Reference To LREAL;
  aCameraMatrix : Reference To TcVnMatrix3x3_LREAL;
  aDistortionCoefficients : Reference To TcVnArray8_LREAL;
  aRotationMatrix : Reference To TcVnMatrix3x3_LREAL;
  aTranslationVector : Reference To TcVnVector3_LREAL;
  fZ : LREAL;
  hrPrev : HRESULT;
END_VAR
```

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aImagePoint1</td>
<td>Reference To TcVnPoint2_REAL</td>
<td>First point in the image coordinate system</td>
</tr>
<tr>
<td>aImagePoint2</td>
<td>Reference To TcVnPoint2_REAL</td>
<td>Second point in the image coordinate system</td>
</tr>
<tr>
<td>fWorldDistance</td>
<td>Reference To LREAL</td>
<td>Returns the distance between aImagePoint1 and aImagePoint2 in the world coordinate system</td>
</tr>
<tr>
<td>aCameraMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL</td>
<td>3x3 camera matrix containing the intrinsic parameters</td>
</tr>
<tr>
<td>aDistortionCoefficients</td>
<td>Reference To TcVnArray8_LREAL</td>
<td>Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]</td>
</tr>
<tr>
<td>aRotationMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL</td>
<td>3x3 rotation matrix</td>
</tr>
<tr>
<td>aTranslationVector</td>
<td>Reference To TcVnVector3_LREAL</td>
<td>Translation vector</td>
</tr>
<tr>
<td>fZ</td>
<td>LREAL</td>
<td>z coordinate (world coordinate system) of the given points (0 would be at (toplevel) calibration pattern)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

#### Return value

`HRESULT`
6.4.12.1.18  F_VN_SortDetectedPatternPoints

Sort the detected pattern points according to the relative positions of the calibration pattern points.

Syntax

Definition:

```
FUNCTION F_VN_SortDetectedPatternPoints : HRESULT
VAR_INPUT
  ipImagePoints : ITcVnContainer;
  ipPatternPoints : ITcVnContainer;
  hrPrev            : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImagePoints</td>
<td>ITcVnContainer</td>
<td>Detected pattern point positions in the image coordinate system (will be sorted by this function, ContainerType_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>ipPatternPoints</td>
<td>ITcVnContainer</td>
<td>The pattern points in the calibration pattern coordinate system, used as a reference to sort ipImagePoints (ContainerType_Vector_TcVnPoint3_REAL)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Metrology 2D

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.12.1.19  F_VN_TransformCoordinatesImageToWorld_Container

Transform 2D image point coordinates to 3D world coordinates (using intrinsic + extrinsic calibration results).

**Syntax**

**Definition:**

```c
FUNCTION F_VN_TransformCoordinatesImageToWorld_Container : HRESULT
VAR_INPUT
   ipSrcPoints2D : ITCvContainer;
   ipDestPoints3D : Reference To ITCvContainer;
   aCameraMatrix : Reference To TcVnMatrix3x3_LREAL;
   aDistortionCoefficients : Reference To TcVnArray8_LREAL;
   aRotationMatrix : Reference To TcVnMatrix3x3_LREAL;
   aTranslationVector : Reference To TcVnVector3_LREAL;
   fZ : LREAL;
   hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcPoints2D</td>
<td>ITCvContainer [345]</td>
<td>Container with 2D source points (TcVnPoint2_DINT or TcVnPoint2_REAL or TcVnPoint2_LREAL)</td>
</tr>
<tr>
<td>ipDestPoints3D</td>
<td>Reference To ITCvContainer [345]</td>
<td>Returns the transformed 3D points (TcVnPoint3_LREAL)</td>
</tr>
<tr>
<td>aCameraMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>3x3 camera matrix containing the intrinsic parameters</td>
</tr>
<tr>
<td>aDistortionCoefficients</td>
<td>Reference To TcVnArray8_LREAL [153]</td>
<td>Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]</td>
</tr>
<tr>
<td>aRotationMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>3x3 rotation matrix</td>
</tr>
<tr>
<td>aTranslationVector</td>
<td>Reference To TcVnVector3_LREAL [153]</td>
<td>Translation vector</td>
</tr>
<tr>
<td>fZ</td>
<td>LREAL</td>
<td>z coordinate (world coordinate system) of the transformed points (0 would be at (toplevel) calibration pattern)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Metrology 2D
### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.12.1.20  F_VN_TransformCoordinatesImageToWorld_Point

**Function:**

Transform 2D image point coordinate to 3D world coordinate (using intrinsic + extrinsic calibration results).

**Syntax**

```delphi
FUNCTION F_VN_TransformCoordinatesImageToWorld_Point : HRESULT
VAR_INPUT
  aSrcPoint       : Reference To TcVnPoint2_LREAL;
  aDestPoint      : Reference To TcVnPoint3_LREAL;
  aCameraMatrix   : Reference To TcVnMatrix3x3_LREAL;
  aDistortionCoefficients : Reference To TcVnArray8_LREAL;
  aRotationMatrix : Reference To TcVnMatrix3x3_LREAL;
  aTranslationVector : Reference To TcVnVector3_LREAL;
  fZ               : LREAL;
  hrPrev           : HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aSrcPoint</td>
<td>Reference To TcVnPoint2_LREAL [151]</td>
<td>Point in the image coordinate system</td>
</tr>
<tr>
<td>aDestPoint</td>
<td>Reference To TcVnPoint3_LREAL [151]</td>
<td>Returns the point in the world coordinate system</td>
</tr>
<tr>
<td>aCameraMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>3x3 camera matrix containing the intrinsic parameters</td>
</tr>
<tr>
<td>aDistortionCoefficients</td>
<td>Reference To TcVnArray8_LREAL [153]</td>
<td>Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]</td>
</tr>
<tr>
<td>aRotationMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>3x3 rotation matrix</td>
</tr>
<tr>
<td>aTranslationVector</td>
<td>Reference To TcVnVector3_LREAL [153]</td>
<td>Translation vector</td>
</tr>
<tr>
<td>fZ</td>
<td>LREAL</td>
<td>z coordinate (world coordinate system) of the transformed points (0 would be at (toplevel) calibration pattern)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Required License
TC3 Vision Metrology 2D

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.12.1.21  F_VN_TransformCoordinatesPlanar_Container

Compute real-world coordinates for given image points or vice versa.

Syntax

Definition:

FUNCTION F_VN_TransformCoordinatesPlanar_Container : HRESULT
VAR_INPUT
  ipSrcPoints : ITCvNContainer;
  ipDestPoints : Reference To ITCvNContainer;
  aTransformationMatrix : Reference To TcVnMatrix3x3_LREAL;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcPoints</td>
<td>ITCvNContainer [345]</td>
<td>Container with 2D source points (TcVnPoint2_DINT or TcVnPoint2_REAL or TcVnPoint2_LREAL)</td>
</tr>
<tr>
<td>ipDestPoints</td>
<td>Reference To ITCvNContainer</td>
<td>Returns the transformed 2D points (TcVnPoint2_LREAL)</td>
</tr>
<tr>
<td>aTransformationMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>Transformation matrix obtained with VnCalibrateCameraPlanar() or its inverse matrix obtained with VnInvertMatrix3x3()</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License
TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.12.1.22 F_VN_TransformCoordinatesPlanar_Point

Compute real-world coordinates for a given image point or vice versa.

Syntax

Definition:

FUNCTION F_VN_TransformCoordinatesPlanar_Point : HRESULT
VAR_INPUT
  aSrcPoint : Reference To TcVnPoint2_LREAL;
  aDestPoint : Reference To TcVnPoint2_LREAL;
  aTransformationMatrix : Reference To TcVnMatrix3x3_LREAL;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aSrcPoint</td>
<td>Reference To TcVnPoint2_LREAL</td>
<td>Point in the source coordinate system</td>
</tr>
<tr>
<td>aDestPoint</td>
<td>Reference To TcVnPoint2_LREAL</td>
<td>Returns point in the destination coordinate system</td>
</tr>
<tr>
<td>aTransformationMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL</td>
<td>Transformation matrix obtained with VnCalibrateCameraPlanar() or its inverse matrix obtained with VnInvertMatrix3x3()</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.12.1.23  F_VN_TransformCoordinatesWorldToImage_Container

Transform 3D world coordinates to 2D image coordinates (using intrinsic + extrinsic calibration results).

**Syntax**

**Definition:**

FUNCTION F_VN_TransformCoordinatesWorldToImage_Container : HRESULT

VAR_INPUT

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcPoints3D</td>
<td>ITCvContainer</td>
<td>Container with 3D source points (TcVnPoint3_LREAL)</td>
</tr>
<tr>
<td>ipDestPoints2D</td>
<td>Reference To ITCvContainer</td>
<td>Returns the transformed 2D points (TcVnPoint2_REAL or TcVnPoint2_DINT, depending on bInteger)</td>
</tr>
<tr>
<td>aCameraMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL</td>
<td>3x3 camera matrix containing the intrinsic parameters</td>
</tr>
<tr>
<td>aDistortionCoefficients</td>
<td>Reference To TcVnArray8_LREAL</td>
<td>Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]</td>
</tr>
<tr>
<td>aRotationMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL</td>
<td>3x3 rotation matrix</td>
</tr>
<tr>
<td>aTranslationVector</td>
<td>Reference To TcVnVector3_LREAL</td>
<td>Translation vector</td>
</tr>
<tr>
<td>bInteger</td>
<td>BOOL</td>
<td>Selects whether the function returns a container of TcVnPoint2_DINT or TcVnPoint2_REAL</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Metrology 2D
6.4.12.1.24  F_VN_TransformCoordinatesWorldToImage_Point

Transform 3D world coordinates to 2D image coordinates (using intrinsic + extrinsic calibration results).

**Syntax**

**Definition:**

FUNCTION F_VN_TransformCoordinatesWorldToImage_Point : HRESULT

VAR_INPUT
  aSrcPoint     Reference To TcVnPoint3_LREAL;
  aDestPoint    Reference To TcVnPoint2_REAL;
  aCameraMatrix Reference To TcVnMatrix3x3_LREAL;
  aDistortionCoefficients Reference To TcVnArray8_LREAL;
  aRotationMatrix Reference To TcVnMatrix3x3_LREAL;
  aTranslationVector Reference To TcVnVector3_LREAL;
  hrPrev        HRESULT;

END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aSrcPoint</td>
<td>Reference To TcVnPoint3_LREAL</td>
<td>3D source point (world coordinate system)</td>
</tr>
<tr>
<td>aDestPoint</td>
<td>Reference To TcVnPoint2_REAL</td>
<td>Returns the transformed 2D point (image coordinate system)</td>
</tr>
<tr>
<td>aCameraMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL</td>
<td>3x3 camera matrix containing the intrinsic parameters</td>
</tr>
<tr>
<td>aDistortionCoefficients</td>
<td>Reference To TcVnArray8_LREAL</td>
<td>Lens distortion coefficients ([k_1, k_2, p_1, p_2, k_3, k_4, k_5, k_6])</td>
</tr>
<tr>
<td>aRotationMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL</td>
<td>3x3 rotation matrix</td>
</tr>
<tr>
<td>aTranslationVector</td>
<td>Reference To TcVnVector3_LREAL</td>
<td>Translation vector</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]
Required License
TC3 Vision Metrology 2D

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.12.2 F_VN_ApplyRotationToAffineTransformation

Apply a rotation to an existing 2D affine transformation matrix.

Syntax

**Definition:**

FUNCTION F_VN_ApplyRotationToAffineTransformation : HRESULT

VAR_INPUT

- aTransformationMatrix : Reference To TcVnMatrix2x3_LREAL;
- fAngle : LREAL;
- hrPrev : HRESULT;

END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aTransformationMatrix</td>
<td>Reference To TcVnMatrix2x3_LREAL</td>
<td>Affine transformation matrix to which the rotation is applied</td>
</tr>
<tr>
<td>fAngle</td>
<td>LREAL</td>
<td>Angle in radians (positive means counter-clockwise)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.12.3 F_VN_ApplyRotationToAffineTransformationExp

Apply a rotation to an existing 2D affine transformation matrix. (expert function)

Syntax

Definition:

FUNCTION F_VN_ApplyRotationToAffineTransformationExp : HRESULT
VAR_INPUT
  aTransformationMatrix : Reference To TcVnMatrix2x3_LREAL;
  fAngle : LREAL;
  bUsePreMultiplication : BOOL;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aTransformationMatrix</td>
<td>Reference To TcVnMatrix2x3_LREAL</td>
<td>Affine transformation matrix to which the rotation is applied</td>
</tr>
<tr>
<td>fAngle</td>
<td>LREAL</td>
<td>Angle in radians (positive means counter-clockwise)</td>
</tr>
<tr>
<td>bUsePreMultiplication</td>
<td>BOOL</td>
<td>Select if pre- or post-multiplication should be used</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
FUNCTION F_VN_ApplyScalingToAffineTransformation : HRESULT
VAR_INPUT
  aTransformationMatrix : Reference To TcVnMatrix2x3_LREAL;
  fScaleX              : LREAL;
  fScaleY              : LREAL;
  hrPrev               : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aTransformationMatrix</td>
<td>Reference To TcVnMatrix2x3_LREAL</td>
<td>Affine transformation matrix to which the scaling is applied</td>
</tr>
<tr>
<td>fScaleX</td>
<td>LREAL</td>
<td>Scaling factor in x direction</td>
</tr>
<tr>
<td>fScaleY</td>
<td>LREAL</td>
<td>Scaling factor in y direction</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.12.5 F_VN_ApplyScalingToAffineTransformationExp

Apply a scaling to an existing 2D affine transformation matrix. (expert function)

Syntax

Definition:

FUNCTION F_VN_ApplyScalingToAffineTransformationExp : HRESULT
VAR_INPUT
  aTransformationMatrix : Reference To TcVnMatrix2x3_LREAL;
  fScaleX              : LREAL;
  fScaleY              : LREAL;
  bUsePreMultiplication : BOOL;
  hrPrev               : HRESULT;
END_VAR
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aTransformationMatrix</td>
<td>Reference To TcVnMatrix2x3 LREAL</td>
<td></td>
</tr>
<tr>
<td>fScaleX</td>
<td>LREAL</td>
<td>Scaling factor in x direction</td>
</tr>
<tr>
<td>fScaleY</td>
<td>LREAL</td>
<td>Scaling factor in y direction</td>
</tr>
<tr>
<td>bUsePreMultiplication</td>
<td>BOOL</td>
<td>Select if pre- or post-multiplication should be used</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.12.6 F_VN_ApplyTranslationToAffineTransformation**

Apply a translation to an existing 2D affine transformation matrix.

**Syntax**

Definition:

FUNCTION F_VN_ApplyTranslationToAffineTransformation : HRESULT
VAR_INPUT
  aTransformationMatrix : Reference To TcVnMatrix2x3 LREAL;
  fDeltaX               : LREAL;
  fDeltaY               : LREAL;
  hrPrev                : HRESULT;
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aTransformationMatrix</td>
<td>Reference To TcVnMatrix2x3_LREAL</td>
<td>Affine transformation matrix to which the translation is applied</td>
</tr>
<tr>
<td>fDeltaX</td>
<td>LREAL</td>
<td>Translation in x direction</td>
</tr>
<tr>
<td>fDeltaY</td>
<td>LREAL</td>
<td>Translation in y direction</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.12.7 F_VN_ApplyTranslationToAffineTransformationExp

Apply a translation to an existing 2D affine transformation matrix. (expert function)

### Syntax

**Definition:**

```c
FUNCTION F_VN_ApplyTranslationToAffineTransformationExp : HRESULT
VAR_INPUT
 TransformationMatrix : Reference To TcVnMatrix2x3_LREAL;
  fDeltaX                  : LREAL;
  fDeltaY                  : LREAL;
  bUsePreMultiplication    : BOOL;
  hrPrev                   : HRESULT;
END_VAR
```
API reference

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aTransformationMatrix</td>
<td>Reference To TcVnMatrix2x3_LREAL</td>
<td>Affine transformation matrix to which the translation is applied</td>
</tr>
<tr>
<td>fDeltaX</td>
<td>LREAL</td>
<td>Translation in x direction</td>
</tr>
<tr>
<td>fDeltaY</td>
<td>LREAL</td>
<td>Translation in y direction</td>
</tr>
<tr>
<td>bUsePreMultiplication</td>
<td>BOOL</td>
<td>Select if pre- or post-multiplication should be used</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [{135}]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [{135}]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.12.8 **F_VN_ApplyYAxisInversionToAffineTransformation**

Apply an inversion of the y-axis direction to an existing 2D affine transformation matrix.

**Syntax**

```plaintext
FUNCTION F_VN_ApplyYAxisInversionToAffineTransformation : HRESULT
VAR_INPUT
  aTransformationMatrix : Reference To TcVnMatrix2x3_LREAL;
  hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aTransformationMatrix</td>
<td>Reference To TcVnMatrix2x3_LREAL</td>
<td>Affine transformation matrix to which the inversion of the y-axis direction is applied</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [{135}]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [{135}]
6.4.12.9  F_VN_ApplyYAxisInversionToAffineTransformationExp

Apply an inversion of the y-axis direction to an existing 2D affine transformation matrix. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_ApplyYAxisInversionToAffineTransformationExp : HRESULT
VAR_INPUT
    aTransformationMatrix : Reference To TcVnMatrix2x3_LREAL;
    bUsePreMultiplication : BOOL;
    hrPrev               : HRESULT;
END_VAR
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aTransformationMatrix</td>
<td>Reference To TcVnMatrix2x3_LREAL</td>
<td>Affine transformation matrix to which the inversion of the y-axis direction is applied</td>
</tr>
<tr>
<td>bUsePreMultiplication</td>
<td>BOOL</td>
<td>Select if pre- or post-multiplication should be used</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

**HRESULT [135]**

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64)</td>
<td>Tc3_Vision</td>
</tr>
<tr>
<td></td>
<td>with PL50, e.g.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Intel 4-core Atom CPU</td>
<td></td>
</tr>
</tbody>
</table>
6.4.12.10  F_VN_ConvertCartesianToPolarAngleImage

Converts cartesian coordinates (x, y) to polar angle.

**Syntax**

**Definition:**

FUNCTION F_VN_ConvertCartesianToPolarAngleImage : HRESULT

VAR_INPUT

  ipSrcImageX : ITCvImage;
  ipSrcImageY : ITCvImage;
  ipDestImageAng : Reference To ITCvImage;
  hrPrev : HRESULT;

END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImageX</td>
<td>ITCvImage [P.383]</td>
<td>Source image containing the x values (elements of type REAL or LREAL)</td>
</tr>
<tr>
<td>ipSrcImageY</td>
<td>ITCvImage [P.383]</td>
<td>Source image containing the y values (Same element type as ipSrcImageX)</td>
</tr>
<tr>
<td>ipDestImageAng</td>
<td>Reference To ITCvImage [P.383]</td>
<td>Destination image containing the angles (Same element type as source images. An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [P.135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [P.135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.12.11  F_VN_ConvertCartesianToPolarAngleImageExp

Converts cartesian coordinates (x, y) to polar angle. (expert function)
Syntax

Definition:

FUNCTION F_VN_ConvertCartesianToPolarAngleImageExp : HRESULT
VAR_INPUT
    ipSrcImageX : ITCvImage;
ipSrcImageY : ITCvImage;
ipDestImageAng : Reference To ITCvImage;
bAngleInDegrees : BOOL;
hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImageX</td>
<td>ITCvImage[383]</td>
<td>Source image containing the x values (elements of type REAL or LREAL)</td>
</tr>
<tr>
<td>ipSrcImageY</td>
<td>ITCvImage[383]</td>
<td>Source image containing the y values (Same element type as ipSrcImageX)</td>
</tr>
<tr>
<td>ipDestImageAng</td>
<td>Reference To ITCvImage[383]</td>
<td>Destination image containing the angles (Same element type as source images. An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>bAngleInDegrees</td>
<td>BOOL</td>
<td>Specifies, if the angles should be in degrees or radians</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT[135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.12.12 F_VN_ConvertCartesianToPolarAngles

Converts cartesian coordinates (x, y) to polar angle.

Syntax

Definition:

FUNCTION F_VN_ConvertCartesianToPolarAngles : HRESULT
VAR_INPUT
    ipSrcPointsCart : ITCvContainer;
ipDestAngles : Reference To ITCvContainer;
hrPrev : HRESULT;
END_VAR
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcPointsCart</td>
<td>ITcVnContainer [345]</td>
<td>Container with cartesian points (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)</td>
</tr>
<tr>
<td>ipDestAngles</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container which is filled with the angles (ContainerType_Vector_REAL, if source points are of type TcVnPoint2_DINT or TcVnPoint2_REAL; else ContainerType_Vector_LREAL. Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g.</td>
<td>Tc3_Vision</td>
</tr>
<tr>
<td></td>
<td>Intel 4-core Atom CPU</td>
<td></td>
</tr>
</tbody>
</table>

6.4.12.13 F_VN_ConvertCartesianToPolarAnglesExp

Converts cartesian coordinates (x, y) to polar angle. (expert function)

Syntax

Definition:

FUNCTION F_VN_ConvertCartesianToPolarAnglesExp : HRESULT
VAR_INPUT
    ipSrcPointsCart : ITcVnContainer;
    ipDestAngles : Reference To ITcVnContainer;
    bAngleInDegrees : BOOL;
    hrPrev : HRESULT;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcPointsCart</td>
<td>ITcVnContainer</td>
<td>Container with cartesian points (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)</td>
</tr>
<tr>
<td>ipDestAngles</td>
<td>Reference To ITcVnContainer</td>
<td>Returns a container which is filled with the angles (ContainerType_Vector_REAL, if source points are of type TcVnPoint2_DINT or TcVnPoint2_REAL; else ContainerType_Vector_LREAL. Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>bAngleInDegrees</td>
<td>BOOL</td>
<td>Specify, if the angles should be in degrees or radians</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.12.14 F_VN_ConvertCartesianToPolarImages

```plaintext
F_VN_ConvertCartesianToPolarImages
```

Converts cartesian coordinates \((x, y)\) to polar coordinates \((\text{magnitude}, \text{angle})\).

### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_ConvertCartesianToPolarImages : HRESULT
VAR_INPUT
    ipSrcImageX : ITcVnImage;
    ipSrcImageY : ITcVnImage;
    ipDestImageMag : Reference To ITcVnImage;
    ipDestImageAng : Reference To ITcVnImage;
    hrPrev : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImageX</td>
<td>ITcVnImage</td>
<td>Source image containing the x values (elements of type REAL or LREAL)</td>
</tr>
<tr>
<td>ipSrcImageY</td>
<td>ITcVnImage</td>
<td>Source image containing the y values (Same element type as ipSrcImageX)</td>
</tr>
<tr>
<td>ipDestImageMag</td>
<td>Reference To ITcVnImage</td>
<td>Destination image containing the magnitudes (Same element type as source images. An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>ipDestImageAng</td>
<td>Reference To ITcVnImage</td>
<td>Destination image containing the angles (Same element type as source images. An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.12.15  F_VN_ConvertCartesianToPolarImagesExp

### Description:

Converts cartesian coordinates \((x, y)\) to polar coordinates \((\text{magnitude}, \angle)\). (expert function)

### Syntax

```plaintext
FUNCTION F_VN_ConvertCartesianToPolarImagesExp : HRESULT
VAR_INPUT
    ipSrcImageX : ITcVnImage;
    ipSrcImageY : ITcVnImage;
    ipDestImageMag : Reference To ITcVnImage;
    ipDestImageAng : Reference To ITcVnImage;
    bAngleInDegrees : BOOL;
    hrPrev : HRESULT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImageX</td>
<td>ITcVnImage [383]</td>
<td>Source image containing the x values (elements of type REAL or LREAL)</td>
</tr>
<tr>
<td>ipSrcImageY</td>
<td>ITcVnImage [383]</td>
<td>Source image containing the y values (Same element type as ipSrcImageX)</td>
</tr>
<tr>
<td>ipDestImageMag</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image containing the magnitudes (Same element type as source images. An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>ipDestImageAn</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image containing the angles (Same element type as source images. An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>bAngleInDegrees</td>
<td>BOOL</td>
<td>Specifies, if the angles should be in degrees or radians</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.12.16 **F_VN_ConvertCartesianToPolarMagnitudeImage**

_F_VN_ConvertCartesianToPolarMagnitudeImage_.

**Syntax**

**Definition:**

FUNCTION _F_VN_ConvertCartesianToPolarMagnitudeImage_ : HRESULT

VAR_INPUT

  - ipSrcImageX : ITcVnImage;
  - ipSrcImageY : ITcVnImage;
  - ipDestImageMag : Reference To ITcVnImage;
  - hrPrev : HRESULT;

END_VAR

Converts cartesian coordinates (x, y) to polar magnitude.

**Syntax**

**Definition:**

FUNCTION _F_VN_ConvertCartesianToPolarMagnitudeImage_ : HRESULT

VAR_INPUT

  - ipSrcImageX : ITcVnImage;
  - ipSrcImageY : ITcVnImage;
  - ipDestImageMag : Reference To ITcVnImage;
  - hrPrev : HRESULT;

END_VAR

Converts cartesian coordinates (x, y) to polar magnitude.
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImageX</td>
<td>ITcVnImage [383]</td>
<td>Source image containing the x values (elements of type REAL or LREAL)</td>
</tr>
<tr>
<td>ipSrcImageY</td>
<td>ITcVnImage [383]</td>
<td>Source image containing the y values (Same element type as ipSrcImageX)</td>
</tr>
<tr>
<td>ipDestImageM</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image containing the magnitudes (Same element type as source images. An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.12.17  F_VN_ConvertCartesianToPolarMagnitudes

Converts cartesian coordinates (x, y) to polar magnitude.

#### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_ConvertCartesianToPolarMagnitudes : HRESULT
VAR_INPUT
    ipSrcPointsCart : ITcVnContainer;
    ipDestMags      : Reference To ITcVnContainer;
    hrPrev           : HRESULT;
END_VAR
```
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcPointsCart</td>
<td>ITcVnContainer [135]</td>
<td>Container with cartesian points (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)</td>
</tr>
<tr>
<td>ipDestMags</td>
<td>Reference To ITcVnContainer [135]</td>
<td>Returns a container which is filled with the magnitudes (ContainerType_Vector_REAL, if source points are of type TcVnPoint2_DINT or TcVnPoint2_REAL; else ContainerType_Vector_LREAL. Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.12.18 F_VN_ConvertCartesianToPolarPoints

Converts cartesian coordinates (x, y) to polar coordinates (magnitude, angle).

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_ConvertCartesianToPolarPoints : HRESULT
VAR_INPUT
    ipSrcPointsCart : ITcVnContainer;
    ipDestPointsPolar : Reference To ITcVnContainer;
    hrPrev : HRESULT;
END_VAR
```
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcPointsCart</td>
<td>ITcVnContainer [345]</td>
<td>Container with cartesian points (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)</td>
</tr>
<tr>
<td>ipDestPointsPolar</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container which is filled with the converted points in polar coordinates (ContainerType_Vector_TcVnPoint2_REAL, if source points are of type TcVnPoint2_DINT or TcVnPoint2_REAL; else ContainerType_Vector_TcVnPoint2_LREAL. Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.12.19  F_VN_ConvertCartesianToPolarPointsExp

Converts cartesian coordinates (x, y) to polar coordinates (magnitude, angle). (expert function)

Syntax

Definition:

FUNCTION F_VN_ConvertCartesianToPolarPointsExp : HRESULT
VAR_INPUT
    ipSrcPointsCart     : ITcVnContainer;
    ipDestPointsPolar   : Reference To ITcVnContainer;
    bAngleInDegrees     : BOOL;
    hrPrev              : HRESULT;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcPointsCart</td>
<td>ITcVnContainer [345]</td>
<td>Container with cartesian points (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)</td>
</tr>
<tr>
<td>ipDestPointsPolar</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container which is filled with the converted points in polar coordinates (ContainerType_Vector_TcVnPoint2_REAL, if source points are of type TcVnPoint2_DINT or TcVnPoint2_REAL; else ContainerType_Vector_TcVnPoint2_LREAL. Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>bAngleInDegrees</td>
<td>BOOL</td>
<td>Specifies, if the angles should be in degrees or radians</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## Syntax

### 6.4.12.20 F_VN_ConvertPolarToCartesianImages

Converts polar coordinates (magnitude, angle) to cartesian coordinates (x, y).

**Definition:**

```c
FUNCTION F_VN_ConvertPolarToCartesianImages : HRESULT
VAR_INPUT
    ipSrcImageMag : ITcVnImage;
    ipSrcImageAng : ITcVnImage;
    ipDestImageX : Reference To ITcVnImage;
    ipDestImageY : Reference To ITcVnImage;
    hrPrev        : HRESULT;
END_VAR
```
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImageMag</td>
<td>ITcVnImage [383]</td>
<td>Source image containing the magnitudes (elements of type REAL or LREAL)</td>
</tr>
<tr>
<td>ipSrcImageAng</td>
<td>ITcVnImage [383]</td>
<td>Source image containing the angles (Same element type as ipSrcImageMag)</td>
</tr>
<tr>
<td>ipDestImageX</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image containing the x values (Same element type as source images. An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>ipDestImageY</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image containing the y values (Same element type as source images. An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

#### 6.4.12.21 F_VN_ConvertPolarToCartesianImagesExp

Converts polar coordinates (magnitude, angle) to cartesian coordinates (x, y). (expert function)

### Syntax

**Definition:**

```
FUNCTION F_VN_ConvertPolarToCartesianImagesExp : HRESULT
VAR_INPUT
    ipSrcImageMag : ITcVnImage;
    ipSrcImageAng : ITcVnImage;
    ipDestImageX  : Reference To ITcVnImage;
    ipDestImageY  : Reference To ITcVnImage;
    bAngleInDegrees : BOOL;
    hrPrev : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImageMag</td>
<td>ITCvImage[383]</td>
<td>Source image containing the magnitudes (elements of type REAL or LREAL)</td>
</tr>
<tr>
<td>ipSrcImageAng</td>
<td>ITCvImage[383]</td>
<td>Source image containing the angles (Same element type as ipSrcImageMag)</td>
</tr>
<tr>
<td>ipDestImageX</td>
<td>Reference To ITCvImage[383]</td>
<td>Destination image containing the x values (Same element type as source images. An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>ipDestImageY</td>
<td>Reference To ITCvImage[383]</td>
<td>Destination image containing the y values (Same element type as source images. An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>bAngleInDegrees</td>
<td>BOOL</td>
<td>Specifies, if the angles are in degrees or radians</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT[135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.12.22 F_VN_ConvertPolarToCartesianPoints

Converts polar coordinates (magnitude, angle) to cartesian coordinates (x, y).

### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_ConvertPolarToCartesianPoints : HRESULT
VAR_INPUT
  ipSrcPointsPolar : ITCvImage;
  ipDestPointsCart : Reference To ITCvImage;
  hrPrev            : HRESULT;
END_VAR
```

```plaintext
F_VN_ConvertPolarToCartesianPoints
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcPointsPolar</td>
<td>ITcVnContainer [345]</td>
<td>Container with polar coordinates (ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)</td>
</tr>
<tr>
<td>ipDestPointsCart</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container which is filled with the converted points in cartesian coordinates (same type ID as ipSrcPointsPolar. Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.12.23 F_VN_ConvertPolarToCartesianPointsExp

Converting polar coordinates (magnitude, angle) to cartesian coordinates (x, y). (expert function)

### Syntax

**Definition:**

```c
FUNCTION F_VN_ConvertPolarToCartesianPointsExp : HRESULT
VAR_INPUT
   ipSrcPointsPolar : ITcVnContainer;
   ipDestPointsCart : Reference To ITcVnContainer;
   bAngleInDegrees : BOOL;
   hrPrev : HRESULT;
END_VAR
```
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcPointsPolar</td>
<td>ITcVnContainer [345]</td>
<td>Container with polar coordinates (ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)</td>
</tr>
<tr>
<td>ipDestPointsCart</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container which is filled with the converted points in cartesian coordinates (same type ID as ipSrcPointsPolar. Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>bAngleInDegrees</td>
<td>BOOL</td>
<td>Specifies, if the angles are in degrees or radians</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.12.24  F_VN_DecomposeHomography

F_VN_DecomposeHomography

Decompose a homography matrix into four solutions of rotation, translation, and plane normal. It returns two solutions (Ra, Ta, Nb) and (Rb, Tb, Nb). The third and the fourth solutions can be calculated as follows: (Ra, -Ta, -Nb) and (Rb, -Tb, -Nb).

Syntax

Definition:

```
FUNCTION F_VN_DecomposeHomography : HRESULT
VAR_INPUT
  aHomography : Reference To TcVnMatrix3x3_LREAL;
  aCameraMatrix : Reference To TcVnMatrix3x3_LREAL;
  aRotationMatrixA : Reference To TcVnMatrix3x3_LREAL;
  aTranslationVectorA : Reference To TcVnVector3_LREAL;
  aNormVectorA : Reference To TcVnVector3_LREAL;
  aRotationMatrixB : Reference To TcVnMatrix3x3_LREAL;
  aTranslationVectorB : Reference To TcVnVector3_LREAL;
  aNormVectorB : Reference To TcVnVector3_LREAL;
  hrPrev : HRESULT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aHomography</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>Homography matrix (perspective transformation matrix)</td>
</tr>
<tr>
<td>aCameraMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>Camera matrix</td>
</tr>
<tr>
<td>aRotationMatrixA</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>Rotation matrix of the first solution (Ra)</td>
</tr>
<tr>
<td>aTranslationVectorA</td>
<td>Reference To TcVnVector3_LREAL [153]</td>
<td>Translation vector of the first solution (Ta)</td>
</tr>
<tr>
<td>aNormVectorA</td>
<td>Reference To TcVnVector3_LREAL [153]</td>
<td>Norm vector of the first solution (Na)</td>
</tr>
<tr>
<td>aRotationMatrixB</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>Rotation matrix of the second solution (Rb)</td>
</tr>
<tr>
<td>aTranslationVectorB</td>
<td>Reference To TcVnVector3_LREAL [153]</td>
<td>Translation vector of the second solution (Tb)</td>
</tr>
<tr>
<td>aNormVectorB</td>
<td>Reference To TcVnVector3_LREAL [153]</td>
<td>Norm vector of the second solution (Nb)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nSolutions</td>
<td>UDINT</td>
<td>Returns the number of solutions. In normal cases, it is equal four.</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.12.25 \textbf{F\_VN\_DecomposeHomographyExp}

Decompose a homography matrix and verify four solutions of rotation, translation, and plane normal. It uses visible reference points being in front of the camera to confirm which solution(s) of the four solutions (maximum two solutions) are consistent with all reference points.

\textbf{Syntax}

\textbf{Definition:}

\begin{verbatim}
FUNCTION F_VN_DecomposeHomographyExp : HRESULT
VAR_INPUT
    aHomography            : Reference To TcVnMatrix3x3_LREAL;
    aCameraMatrix          : Reference To TcVnMatrix3x3_LREAL;
    aRotationMatrixA       : Reference To TcVnMatrix3x3_LREAL;
    aTranslationVectorA    : Reference To TcVnVector3_LREAL;
    aNormVectorA           : Reference To TcVnVector3_LREAL;
    aTranslationVectorB    : Reference To TcVnVector3_LREAL;
    aRotationMatrixB       : Reference To TcVnMatrix3x3_LREAL;
    aNormVectorB           : Reference To TcVnVector3_LREAL;
    ipPointsSrc            : ITCvNContainer;
    ipPointsDes            : ITCvNContainer;
    ipInlierMask           : ITCvNContainer;
    hrPrev                 : HRESULT;
END_VAR
VAR_OUTPUT
    nSolutions             : UDINT;
END_VAR
\end{verbatim}
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aHomography</td>
<td>Reference To TcVnMatrix3x3_LREAL</td>
<td>Homography matrix (perspective transformation matrix)</td>
</tr>
<tr>
<td>aCameraMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL</td>
<td>Camera matrix</td>
</tr>
<tr>
<td>aRotationMatrixA</td>
<td>Reference To TcVnMatrix3x3_LREAL</td>
<td>Rotation matrix of the first solution (Ra)</td>
</tr>
<tr>
<td>aTranslationVectorA</td>
<td>Reference To TcVnVector3_LREAL</td>
<td>Translation vector of the first solution (Ta)</td>
</tr>
<tr>
<td>aNormVectorA</td>
<td>Reference To TcVnVector3_LREAL</td>
<td>Norm vector of the first solution (Na)</td>
</tr>
<tr>
<td>aRotationMatrixB</td>
<td>Reference To TcVnMatrix3x3_LREAL</td>
<td>Rotation matrix of the second solution (Rb)</td>
</tr>
<tr>
<td>aTranslationVectorB</td>
<td>Reference To TcVnVector3_LREAL</td>
<td>Translation vector of the second solution (Tb)</td>
</tr>
<tr>
<td>aNormVectorB</td>
<td>Reference To TcVnVector3_LREAL</td>
<td>Norm vector of the second solution (Nb)</td>
</tr>
<tr>
<td>ipPointsSrc</td>
<td>ITcVnContainer [135]</td>
<td>Container with source points (ContainerType_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>ipPointsDes</td>
<td>ITcVnContainer [135]</td>
<td>Container with destination points (same number as ipPoints1, ContainerType_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>ipInlierMask</td>
<td>ITcVnContainer [135]</td>
<td>A mask marking the inliers (optional, set to 0 if not available; ContainerType_Vector_SINT or ContainerType_Vector_USINT)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nSolutions</td>
<td>UDINT</td>
<td>Return the number of true potential solutions [0, 1, or 2]. 0: no true potential solutions for the given corresponding points are confirmed. 1: the solution (Ra, Ta, Na) is confirmed. 2: both solution (Ra, Ta, Na) and (Rb, Tb, Nb) are confirmed. In all cases the function returns two solutions (Ra, Ta, Nb) and (Rb, Tb, Nb). The third and the fourth solutions can be calculated as follows: (Ra, -Ta, -Na) and (Rb, -Tb, -Nb).</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base
### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.12.26 F_VN_FlipImage

**F_VN_FlipImage**

```
FUNCTION F_VN_FlipImage : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  eFlipAxis : ETcVnFlipAxis;
  hrPrev : HRESULT;
END_VAR
```

Flip an image (mirror and shift back to prior coordinates).

**Syntax**

**Definition:**

```
FUNCTION F_VN_FlipImage : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  eFlipAxis : ETcVnFlipAxis;
  hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>eFlipAxis</td>
<td>ETcVnFlipAxis [190]</td>
<td>Selects the axis around which to flip the image</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.12.27 F_VN_GenerateAffineTransformationUnitMatrix2D

**F_VN_GenerateAffineTransformationUnitMatrix2D**

```
FUNCTION F_VN_GenerateAffineTransformationUnitMatrix2D : HRESULT
VAR_INPUT
  eTransformationMatrix : Reference To TcVnMatrix2x3_LREAL;
  hrPrev : HRESULT;
END_VAR
```

```
Generate an affine transformation 2D unit matrix.

**Syntax**

**Definition:**

```
FUNCTION F_VN_GenerateAffineTransformationUnitMatrix2D : HRESULT
VAR_INPUT
    aTransformationMatrix : Reference To TcVnMatrix2x3_LREAL;
    hrPrev       : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aTransformationMatrix</td>
<td>Reference To TcVnMatrix2x3_LREAL [153]</td>
<td>Resulting affine transformation unit matrix</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.12.28 **F_VN_GetAffineTransform**

```
F_VN_GetAffineTransform
    aSrcPoints    Reference To TcVnArray3_Point2_REAL
    aDestPoints   Reference To TcVnArray3_Point2_REAL
    aTransformationMatrix Reference To TcVnMatrix2x3_LREAL
    hrPrev        HRESULT
```

Calculate the affine transformation between three corresponding point pairs. The points mark the corners of the corresponding triangles.

**Syntax**

**Definition:**

```
FUNCTION F_VN_GetAffineTransform : HRESULT
VAR_INPUT
    aSrcPoints    : Reference To TcVnArray3_Point2_REAL;
    aDestPoints   : Reference To TcVnArray3_Point2_REAL;
    aTransformationMatrix : Reference To TcVnMatrix2x3_LREAL;
    hrPrev        : HRESULT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aSrcPoints</td>
<td>Reference To TcVnArray3_Point2_REAL</td>
<td>Source points</td>
</tr>
<tr>
<td>aDestPoints</td>
<td>Reference To TcVnArray3_Point2_REAL</td>
<td>Destination points</td>
</tr>
<tr>
<td>aTransformationMatrix</td>
<td>Reference To TcVnMatrix2x3_LREAL</td>
<td>Resulting transformation matrix</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.12.29  F_VN_GetAffineTransformation2D

Estimate the 2D affine transformation between two planar point sets.

**Syntax**

**Definition:**

FUNCTION F_VN_GetAffineTransformation2D : HRESULT
VAR_INPUT
    ipSrcPoints : ITcVnContainer;
    ipDestPoints : ITcVnContainer;
    aAffineTransform : Reference To TcVnMatrix2x3_LREAL;
    hrPrev : HRESULT;
END_VAR
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcPoints</td>
<td>ITcVnContainer [345]</td>
<td>Container with at least 3 source points (ContainerType_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>ipDestPoints</td>
<td>ITcVnContainer [345]</td>
<td>Container with destination points (same number as ipSrcPoints, ContainerType_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>aAffineTransform</td>
<td>Reference To TcVnMatrix2x3_LREAL [153]</td>
<td>Returns the affine transformation matrix, which transforms the source points to the destination points</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.12.30 F_VN_GetAffineTransformation2DExp**

Estimate the affine transformation between two planar point sets. (expert function)

**Syntax**

**Definition:**

FUNCTION F_VN_GetAffineTransformation2DExp : HRESULT
VAR_INPUT
  ipSrcPoints : ITcVnContainer;
  ipDestPoints : ITcVnContainer;
  aAffineTransform : Reference To TcVnMatrix2x3_LREAL;
  eAlgorithm : ETcVnEstimationAlgorithm;
  fReprojThreshold : LREAL;
  ipInlierMask : Reference To ITcVnContainer;
  nMaxIterations : UDINT;
  fConfidence : LREAL;
  nRefineIters : UDINT;
  hrPrev : HRESULT;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcPoints</td>
<td>ITcVnContainer [345]</td>
<td>Container with at least 3 source points (ContainerType_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>ipDestPoints</td>
<td>ITcVnContainer [345]</td>
<td>Container with destination points (same number as ipSrcPoints, ContainerType_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>aAffineTransform</td>
<td>Reference To TcVnMatrix2x3_LREAL [153]</td>
<td>Returns the affine transformation matrix, which transforms the source points to the destination points</td>
</tr>
<tr>
<td>eAlgorithm</td>
<td>ETcVnEstimationAlgorithm [187]</td>
<td>Estimation algorithm (only RANSAC and LMEDS are supported)</td>
</tr>
<tr>
<td>fReprojThreshold</td>
<td>LREAL</td>
<td>Maximum allowed reprojection error to treat a point pair as an inlier</td>
</tr>
<tr>
<td>ipInlierMask</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a mask marking the inliers (optional, set to 0 if not required; ContainerType_Vector_USINT)</td>
</tr>
<tr>
<td>nMaxIterations</td>
<td>UDINT</td>
<td>Maximum number of iterations</td>
</tr>
<tr>
<td>fConfidence</td>
<td>LREAL</td>
<td>Confidence (0..1)</td>
</tr>
<tr>
<td>nRefinements</td>
<td>UDINT</td>
<td>Maximum number of iterations of Levenberg-Marquardt algorithm to refine further the affine transform (using only inliers). Set to 0 to disable refining.</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.12.31 F_VN_GetPerspectiveTransformation

Calculate the perspective transformation between four corresponding point pairs. The points mark the corners of the corresponding rectangles.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_GetPerspectiveTransformation : HRESULT
VAR_INPUT
    aSrcPoints : Reference To TcVnArray4_Point2_REAL;
    aDestPoints : Reference To TcVnArray4_Point2_REAL;
    aTransformationMatrix : Reference To TcVnMatrix3x3_LREAL;
    hrPrev : HRESULT;
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aSrcPoints</td>
<td>Reference To TcVn4_Point2_REAL</td>
<td>Source points</td>
</tr>
<tr>
<td>aDestPoints</td>
<td>Reference To TcVnArrayPoint2_REAL</td>
<td>Destination points</td>
</tr>
<tr>
<td>aTransformationMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL</td>
<td>Resulting transformation matrix</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.12.32 F_VN_Homography**

Find the homography (perspective transformation) between two planar point sets.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_Homography : HRESULT
VAR_INPUT
   ipSrcPoints : ITCvNContainer;
   ipDestPoints : ITCvNContainer;
   aPerspectiveTransform : Reference To TcVnMatrix3x3_LREAL;
   hrPrev : HRESULT;
END_VAR
```
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcPoints</td>
<td>ITcVnContainer [345]</td>
<td>Container with at least 4 source points (ContainerType_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>ipDestPoints</td>
<td>ITcVnContainer [345]</td>
<td>Container with destination points (same number as ipSrcPoints, ContainerType_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>aPerspectiveTransform</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>Returns the perspective transformation matrix, which transforms the source points to the destination points</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.12.33 F_VN_HomographyExp

Find the homography (perspective transformation) between two planar point sets. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_HomographyExp : HRESULT
VAR_INPUT
   ipSrcPoints : ITcVnContainer;
   ipDestPoints : ITcVnContainer;
   aPerspectiveTransform : Reference To TcVnMatrix3x3_LREAL;
   eAlgorithm : ETcVnEstimationAlgorithm;
   fReprojThreshold : LREAL;
   ipInlierMask : Reference To ITcVnContainer;
   nMaxIterations : UDINT;
   fConfidence : LREAL;
   hrPrev : HRESULT;
END_VAR
```
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcPoints</td>
<td>ITcVnContainer [345]</td>
<td>Container with at least 4 source points (ContainerType_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>ipDestPoints</td>
<td>ITcVnContainer [345]</td>
<td>Container with destination points (same number as ipSrcPoints, ContainerType_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>aPerspectiveTr</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>Returns the perspective transformation matrix, which transforms the source points to the destination points</td>
</tr>
<tr>
<td>eAlgorithm</td>
<td>ETcVnEstimationAlgorithm [187]</td>
<td>Estimation algorithm</td>
</tr>
<tr>
<td>fReprojThreshold</td>
<td>LREAL</td>
<td>Maximum allowed reprojection error to treat a point pair as an inlier (only for RANSAC, RHO)</td>
</tr>
<tr>
<td>ipInlierMask</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a mask marking the inliers (optional, set to 0 if not required; ContainerType_Vector_SINT; only for RANSAC, LMEDS)</td>
</tr>
<tr>
<td>nMaxIterations</td>
<td>UDINT</td>
<td>Maximum number of RANSAC iterations</td>
</tr>
<tr>
<td>fConfidence</td>
<td>LREAL</td>
<td>Confidence (0..1)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.12.34 F_VN_InvertAffineTransform

Invert a 2D affine transformation matrix.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_InvertAffineTransform : HRESULT
VAR_INPUT
    aTransformationMatrix : Reference To TcVnMatrix2x3_LREAL;
    aInvertedTransformationMatrix : Reference To TcVnMatrix2x3_LREAL;
    hrPrev : HRESULT;
END_VAR
```

Invert a 2D affine transformation matrix.
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aTransformationMatrix</td>
<td>Reference To TcVnMatrix2x3_LREAL</td>
<td>Source affine transformation matrix</td>
</tr>
<tr>
<td>aInvertedTransformationMatrix</td>
<td>Reference To TcVnMatrix2x3_LREAL</td>
<td>Resulting inverted affine transformation matrix</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.12.35 F_VN_PerspectiveTransformation

Apply a perspective transformation (rotation + translation, e.g. from extrinsic calibration) to 3D point coordinates.

**Syntax**

**Definition:**

FUNCTION F_VN_PerspectiveTransformation : HRESULT
VAR_INPUT
  ipSrcPoints   : ITCvnContainer;
  ipDestPoints  : Reference To ITCvnContainer;
  aRotationMatrix   : Reference To TcVnMatrix3x3_LREAL;
  aTranslationVector : Reference To TcVnVector3_LREAL;
  hrPrev        : HRESULT;
END_VAR

F_VN_PerspectiveTransformation

hrPrev = HRESULT

F_VN_PerspectiveTransformation

hrPrev = HRESULT

F_VN_PerspectiveTransformation
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcPoints</td>
<td>ITcVnContainer [345]</td>
<td>Container with 3D source points (TcVnPoint3_REAL or TcVnPoint3_LREAL)</td>
</tr>
<tr>
<td>ipDestPoints</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns the transformed points (same type as ipSrcPoints)</td>
</tr>
<tr>
<td>aRotationMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>3x3 rotation matrix</td>
</tr>
<tr>
<td>aTranslationVector</td>
<td>Reference To TcVnVector3_LREAL [153]</td>
<td>Translation vector</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.12.36 F_VN_PyramidDown

Downsamples an image to half width and height. Can use available TwinCAT Job Tasks for executing parallel code regions.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_PyramidDown : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.12.37  F_VN_PyramidUp

Upsamples an image to double width and height.

### Syntax

**Definition:**

```c
FUNCTION F_VN_PyramidUp : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev : HRESULT;
END_VAR
```

## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]
Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.12.38  **F_VN_RemapImageToLogPolarSpace**

Remap an image to log-polar space. Can use available TwinCAT Job Tasks for executing parallel code regions.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_RemapImageToLogPolarSpace : HRESULT
VAR_INPUT
  ipSrcImage  : ITCvImage;
  ipDestImage : Reference To ITCvImage;
  aCenter     : Reference To TcVnPoint2_REAL;
  hrPrev      : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITCvImage</td>
<td>Destination image (Must not be the same as ipSrcImage! An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>aCenter</td>
<td>Reference To TcVnPoint2_REAL</td>
<td>Center point for the transformation</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

**HRESULT [135]**

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.12.39  F_VN_RemapImageToLogPolarSpaceExp

Remap an image to log-polar space. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

FUNCTION F_VN_RemapImageToLogPolarSpaceExp : HRESULT
VAR_INPUT
  ipSrcImage : ITCvImage;
  ipDestImage : Reference To ITCvImage;
  aCenter : Reference To TcVnPoint2_REAL;
  fScale : LREAL;
  eInterpolationType : ETcVnInterpolationType;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvImage</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITCvImage</td>
<td>Destination image (Must not be the same as ipSrcImage! An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>aCenter</td>
<td>Reference To TcVnPoint2_REAL</td>
<td>Center point for the transformation</td>
</tr>
<tr>
<td>fScale</td>
<td>LREAL</td>
<td>Magnitude scale parameter (set to 0 for auto select)</td>
</tr>
<tr>
<td>eInterpolationType</td>
<td>ETcVnInterpolationType</td>
<td>Interpolation type</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.12.40 F_VN_RemapImageToPolarSpace

Remap an image to polar space.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

FUNCTION F_VN_RemapImageToPolarSpace : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  aCenter : Reference To TcVnPoint2_REAL;
  hrPrev : HRESULT;
END_VAR

Inputs

Name | Type | Description
--- | --- | ---
ipSrcImage | ITcVnImage | Source image
ipDestImage | Reference To ITcVnImage | Destination image (Must not be the same as ipSrcImage! An appropriate destination image will be created if required.)
aCenter | Reference To TcVnPoint2_REAL | Center point for the transformation
hrPrev | HRESULT | HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [ 135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.12.41 F_VN_RemapImageToPolarSpaceExp
Remap an image to polar space. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax
Definition:

```plaintext
FUNCTION F_VN_RemapImageToPolarSpaceExp : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  aCenter : Reference To TcVnPoint2_REAL;
  fMaxRadius : LREAL;
  eInterpolationType : ETcVnInterpolationType;
  hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (Must not be the same as ipSrcImage! An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>aCenter</td>
<td>Reference To TcVnPoint2_REAL [151]</td>
<td>Center point for the transformation</td>
</tr>
<tr>
<td>fMaxRadius</td>
<td>LREAL</td>
<td>Maximum radius for the transformation (set to 0 for auto select)</td>
</tr>
<tr>
<td>eInterpolationType</td>
<td>ETcVnInterpolationType [192]</td>
<td>Interpolation type</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.12.42  F_VN_ResizeImage

Resize an image using a specific interpolation type.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.
Syntax

Definition:

FUNCTION F_VN_ResizeImage : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    nWidth : UDINT;
    nHeight : UDINT;
    eInterpolationType : ETcVnInterpolationType;
    hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate image will be created if required.)</td>
</tr>
<tr>
<td>nWidth</td>
<td>UDINT</td>
<td>New width</td>
</tr>
<tr>
<td>nHeight</td>
<td>UDINT</td>
<td>New height</td>
</tr>
<tr>
<td>eInterpolationType</td>
<td>ETcVnInterpolationType</td>
<td>Interpolation type</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.12.43   F_VN_RotateImage

Rotate an image by 90, 180, or 270 degrees in clockwise direction.

Syntax

Definition:

FUNCTION F_VN_RotateImage : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    eAngle : ETcVnRotationAngle
    hrPrev : HRESULT
END_VAR
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (An appropriate image will be created if required.)</td>
</tr>
<tr>
<td>eAngle</td>
<td>ETcVnRotationAngle [199]</td>
<td>Rotation angle</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.12.44 F_VN_WarpAffine**

Apply an affine transformation to an image.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

**Syntax**

Definition:

```c
FUNCTION F_VN_WarpAffine : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    aTransformationMatrix : Reference To TcVnMatrix2x3_LREAL;
    hrPrev : HRESULT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate image will be created if required.)</td>
</tr>
<tr>
<td>aTransformationMatrix</td>
<td>Reference To TcVnMatrix2x3_LREAL</td>
<td>Affine transformation matrix</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.12.45 F_VN_WarpAffine_Container**

Apply an affine transformation to a container of 2D points.

**Syntax**

**Definition:**

```
FUNCTION F_VN_WarpAffine_Container : HRESULT
VAR_INPUT
    ipSrcPoints : ITcVnContainer;
    ipDestPoints : Reference To ITcVnContainer;
    aTransformationMatrix : Reference To TcVnMatrix2x3_LREAL;
    hrPrev : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcPoints</td>
<td>ITCvNContainer [345]</td>
<td>Source points (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)</td>
</tr>
<tr>
<td>ipDestPoints</td>
<td>Reference To ITCvNContainer [345]</td>
<td>Returns the transformed points (same type ID as ipSrcPoints; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>aTransformationMatrix</td>
<td>Reference To TcVnMatrix2x3_LREAL [153]</td>
<td>Affine transformation matrix</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.12.46 F_VN_WarpAffine_Point

Apply an affine transformation to a 2D point.

#### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_WarpAffine_Point : HRESULT
VAR_INPUT
    aSrcPoint : Reference To TcVnPoint2_LREAL;
    aDestPoint : Reference To TcVnPoint2_LREAL;
    aTransformationMatrix : Reference To TcVnMatrix2x3_LREAL;
    hrPrev : HRESULT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aSrcPoint</td>
<td>Reference To TcVnPoint2_LREAL [151]</td>
<td>Source point</td>
</tr>
<tr>
<td>aDestPoint</td>
<td>Reference To TcVnPoint2_LREAL [151]</td>
<td>Destination point</td>
</tr>
<tr>
<td>aTransformationMatrix</td>
<td>Reference To TcVnMatrix2x3_LREAL [153]</td>
<td>Affine transformation matrix</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.12.47 F_VN_WarpAffineExp

**Syntax**

Apply an affine transformation to an image. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

**Definition:**

```
FUNCTION F_VN_WarpAffineExp : HRESULT
VAR_INPUT
  ipSrcImage : ITCvImage;
  ipDestImage : Reference To ITCvImage;
  aTransformationMatrix : Reference To TcVnMatrix2x3_LREAL;
  nDestWidth : UDINT;
  nDestHeight : UDINT;
  eInterpolationType : ETcVnInterpolationType;
  bWarpInverse : BOOL;
  eBorderInterpolation : ETcVnBorderInterpolationMethod;
  fBorderValue : REAL;
  hrPrev : HRESULT;
```

```
F_VN_WarpAffineExp
  ipSrcImage ITCvImage
  ipDestImage Reference To ITCvImage
  aTransformationMatrix Reference To TcVnMatrix2x3_LREAL
  nDestWidth UDINT
  nDestHeight UDINT
  eInterpolationType ETcVnInterpolationType
  bWarpInverse BOOL
  eBorderInterpolation ETcVnBorderInterpolationMethod
  fBorderValue REAL
  hrPrev HRESULT
  HRESULT F_VN_WarpAffineExp
```
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (An appropriate image will be created if required.)</td>
</tr>
<tr>
<td>aTransformationMatrix</td>
<td>Reference To TcVnMatrix2x3_LREAL [153]</td>
<td>Affine transformation matrix</td>
</tr>
<tr>
<td>nDestWidth</td>
<td>UDINT</td>
<td>Width of the destination image</td>
</tr>
<tr>
<td>nDestHeight</td>
<td>UDINT</td>
<td>Height of the destination image</td>
</tr>
<tr>
<td>eInterpolationType</td>
<td>ETcVnInterpolationType [192]</td>
<td>Interpolation method</td>
</tr>
<tr>
<td>bWarpInverse</td>
<td>BOOL</td>
<td>Handle aTransformationMatrix as the inverse transformation</td>
</tr>
<tr>
<td>eBorderInterpolation</td>
<td>ETcVnBorderInterpolation Method [157]</td>
<td>Border interpolation method (ISOLATED not supported)</td>
</tr>
<tr>
<td>fBorderValue</td>
<td>LREAL</td>
<td>Border value, if CONSTANT is used</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.12.48 F_VN_WarpPerspective

Apply a perspective transformation to an image.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

### Syntax

Definition:
FUNCTION F_VN_WarpPerspective : HRESULT
VAR_INPUT
    ipSrcImage : ITCvImage;
    ipDestImage : Reference To ITCvImage;
    aTransformationMatrix : Reference To TcVnMatrix3x3_LREAL;
    hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITCvImage [383]</td>
<td>Destination image (An appropriate image will be created if required.)</td>
</tr>
<tr>
<td>aTransformationMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>Perspective transformation matrix</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.12.49 F_VN_WarpPerspective_Container

Apply a perspective transformation to a container of 2D points.

Syntax

Definition:
FUNCTION F_VN_WarpPerspective_Container : HRESULT
VAR_INPUT
    ipSrcPoints : ITCvContainer;
    ipDestPoints : Reference To ITCvContainer;
    aTransformationMatrix : Reference To TcVnMatrix3x3_LREAL;
    hrPrev : HRESULT;
END_VAR

F_VN_WarpPerspective_Container
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcPoints</td>
<td>ITcVnContainer [345]</td>
<td>Source points (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)</td>
</tr>
<tr>
<td>ipDestPoints</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns the transformed points (same type ID as ipSrcPoints; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>aTransformationMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>Perspective transformation matrix</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.12.50 F_VN_WarpPerspective_Point

**F_VN_WarpPerspective_Point**

```
FUNCTION F_VN_WarpPerspective_Point : HRESULT
VAR_INPUT
  aSrcPoint : Reference To TcVnPoint2_LREAL;
  aDestPoint : Reference To TcVnPoint2_LREAL;
  aTransformationMatrix : Reference To TcVnMatrix3x3_LREAL;
  hrPrev : HRESULT;
END_VAR
```

Apply a perspective transformation to a 2D point.

**Syntax**

**Definition:**

FUNCTION F_VN_WarpPerspective_Point : HRESULT
VAR_INPUT
  aSrcPoint : Reference To TcVnPoint2_LREAL;
  aDestPoint : Reference To TcVnPoint2_LREAL;
  aTransformationMatrix : Reference To TcVnMatrix3x3_LREAL;
  hrPrev : HRESULT;
END_VAR
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aSrcPoint</td>
<td>Reference To TcVnPoint2_LREAL [151]</td>
<td>Source point</td>
</tr>
<tr>
<td>aDestPoint</td>
<td>Reference To TcVnPoint2_LREAL [151]</td>
<td>Destination point</td>
</tr>
<tr>
<td>aTransformationMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>Perspective transformation matrix</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.12.51 F_VN_WarpPerspective_Rectangle**

Use a perspective transform to warp a rectangle.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_WarpPerspective_Rectangle : HRESULT
VAR_INPUT
    nTopLeftX : UDINT;
    nTopLeftY : UDINT;
    nBottomRightX : UDINT;
    nBottomRightY : UDINT;
    ipDestPoints : Reference To ITcVnContainer;
    aTransformationMatrix : Reference To TcVnMatrix3x3_LREAL;
    hrPrev : HRESULT;
END_VAR
```
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nTopLeftX</td>
<td>UDINT</td>
<td>x coordinate of the top left corner</td>
</tr>
<tr>
<td>nTopLeftY</td>
<td>UDINT</td>
<td>y coordinate of the top left corner</td>
</tr>
<tr>
<td>nBottomRightX</td>
<td>UDINT</td>
<td>x coordinate of the bottom right corner</td>
</tr>
<tr>
<td>nBottomRightY</td>
<td>UDINT</td>
<td>y coordinate of the bottom right corner</td>
</tr>
<tr>
<td>ipDestPoints</td>
<td>Reference To</td>
<td>Returns the 4 transformed points (ContainerType_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td></td>
<td>ITcVnContainer</td>
<td></td>
</tr>
<tr>
<td>aTransformationMatrix</td>
<td>Reference To</td>
<td>Perspective transformation matrix</td>
</tr>
<tr>
<td></td>
<td>TcVnMatrix3x3_LREAL</td>
<td></td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.12.52 F_VN_WarpPerspectiveExp

Apply a perspective transformation to an image. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

#### Syntax

**Definition:**

```c
FUNCTION F_VN_WarpPerspectiveExp : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    aTransformationMatrix : Reference To TcVnMatrix3x3_LREAL;
    nDestWidth : UDINT;
    nDestHeight : UDINT;
    eInterpolationType : ETcVnInterpolationType;
    bWarpInverse : BOOL;
    fBorderValue : LREAL;
    hrPrev : HRESULT;
RESULT F_VN_WarpPerspectiveExp
```
eBorderInterpolation : ETcVnBorderInterpolationMethod;
fBorderValue : LREAL;
hrPrev : HRESULT;
END_VAR

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (An appropriate image will be created if required.)</td>
</tr>
<tr>
<td>aTransformationMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>Perspective transformation matrix</td>
</tr>
<tr>
<td>nDestWidth</td>
<td>UDINT</td>
<td>Width of the destination image</td>
</tr>
<tr>
<td>nDestHeight</td>
<td>UDINT</td>
<td>Height of the destination image</td>
</tr>
<tr>
<td>eInterpolationType</td>
<td>ETcVnInterpolationType [192]</td>
<td>Interpolation method</td>
</tr>
<tr>
<td>bWarpInverse</td>
<td>BOOL</td>
<td>Handle aTransformationMatrix as the inverse transformation</td>
</tr>
<tr>
<td>eBorderInterpolation</td>
<td>ETcVnBorderInterpolation Method [157]</td>
<td>Border interpolation method (ISOLATED not supported)</td>
</tr>
<tr>
<td>fBorderValue</td>
<td>LREAL</td>
<td>Border value, if CONSTANT is used</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.13 Image Analysis

This group contains functions for extracting information from images.

### Functions

**Object detection [1027]**

- Contour search
- Object finding by Hough transformation
- Matching by image comparison

**Statistical image features**

- F_VN_ImageAverage(Exp) [1066]
• F_VN_ImageAverageStdDev(Exp) [1068]
• F_VN_ImageMedian(Exp) [1072]
• F_VN_MaxPixelValue(Exp) [1075]
• F_VN_MinPixelValue(Exp) [1076]

**Edge detection**
• F_VN_CannyEdgeDetection(Exp) [1055]

**Finding contiguous image regions**
• F_VN_ConnectedComponents(Exp) [1058]
• F_VN_ConnectedComponentsWithStats(Exp) [1061]

**Miscellaneous**
• F_VN_ControlNonZeroPixels [1062]
• F_VN_DistanceTransformation(Exp) [1063]
• F_VN_ImageCenterOfMass(Exp) [1070]
• F_VN_ImageMoments [1074]

**Samples**
Image analysis [1390]

### 6.4.13.1 Object Detection

This subgroup contains functions for object recognition.

**Functions**

**Object detection**
• F_VN_DetectBlobs(Exp) [1029]
• F_VN_FindContours(Exp) [1040]
• F_VN_FindContourHierarchyExp [1033]
• F_VN_HoughCircles(Exp) [1043]
• F_VN_HoughLines(Exp) [1046]
• F_VN_HoughLinesP(Exp) [1048]

**Matching**
• F_VN_MatchImageHuMoments [1049]
• F_VN_MatchTemplate(Exp) [1053]
• F_VN_MatchTemplateAndEvaluate(Exp) [1052]

**Miscellaneous**
• F_VN_AdjustActiveContour [1028]

**Samples**
Object Detection [1390]
6.4.13.1.1  F_VN_AdjustActiveContour

Adjust active contour (snake) in order to minimize its cumulative (internal and external) energy.

Syntax

Definition:

FUNCTION F_VN_AdjustActiveContour : HRESULT
VAR_INPUT
    ipImage : ITCvImage;
    ipActiveContour : ITCvContainer;
    fAlpha : REAL;
    fBeta : REAL;
    fGamma : REAL;
    nWindowWidth : UDINT;
    nWindowHeight : UDINT;
    nMaxIterations : UDINT;
    bUseGradient : BOOL;
    hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage</td>
<td>ITCvImage</td>
<td>Source image (1 channel, USINT elements)</td>
</tr>
<tr>
<td>ipActiveContour</td>
<td>ITCvContainer</td>
<td>Initial contour, which will be adjusted</td>
</tr>
<tr>
<td>fAlpha</td>
<td>REAL</td>
<td>Continuity energy coefficient</td>
</tr>
<tr>
<td>fBeta</td>
<td>REAL</td>
<td>Curvature energy coefficient</td>
</tr>
<tr>
<td>fGamma</td>
<td>REAL</td>
<td>Image energy coefficient</td>
</tr>
<tr>
<td>nWindowWidth</td>
<td>UDINT</td>
<td>Windows width (3, 5, 7, ...)</td>
</tr>
<tr>
<td>nWindowHeight</td>
<td>UDINT</td>
<td>Windows height (3, 5, 7, ...)</td>
</tr>
<tr>
<td>nMaxIterations</td>
<td>UDINT</td>
<td>Maximum iterations</td>
</tr>
<tr>
<td>bUseGradient</td>
<td>BOOL</td>
<td>If true, the gradient magnitude is used as image energy (otherwise: pixel intensity)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT

Required License

TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.13.1.2  F_VN_DetectBlobs

Detects blob-contours. Applies a threshold, a contour search and offers several options for filtering the found contours. Provides easy setup for multiple thresholds and combination of results. Can use available TwinCAT Job Tasks for executing parallel code regions. Can return partial results when canceled by Watchdog.

Syntax

**Definition:**

```plaintext
FUNCTION F_VN_DetectBlobs : HRESULT 
VAR_INPUT
  ipSrcImage : ITCvImage;
  ipBlobContours : Reference To ITCvContainer;
  stParams : Reference To TcVnParamsBlobDetection;
  hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvImage [383]</td>
<td>Source image (USINT or REAL, 1 channel or 3 channel (3 channel input is expected to be RGB and internally converted to Gray))</td>
</tr>
<tr>
<td>ipBlobContours</td>
<td>Reference To ITCvContainer [345]</td>
<td>Returns a container which is filled with the found contours (ContainerType_Vector_Vector_TcVnPoint2_DINT; The elements of this container are single contours. Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>stParams</td>
<td>Reference To TcVnParamsBlobDetection [211]</td>
<td>Parameters to filter the detected contours.</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRRESULT [135]

Further information

The function **F_VN_DetectBlobs** detects blob contours on the basis of definable features in the input image. Internally, this takes place through contour finding analogous to **F_VN_FindContours [1037]** and subsequent filtering of the contours found on the basis of contour features.
Parameter

Input image

The input image `ipSrcImage` must be a 1-channel or 3-channel image of the type `USINT` or `REAL`. 3-channel images are interpreted as RGB and converted internally into grayscale images.

List of contours found (Return value)

All contours found in the input image are returned in the container `ipBlobContours`.

Parameters for blob detection

The parameters `stParams` of the structure `TcVnParamsBlobDetection` define which features a contour in the input image must have in order to be found and returned.

Expert parameters

The expert version `F_VN_DetectBlobsExp` contains additional parameters.

Blobs parameters

Threshold value

In the first step, a contour search is carried out internally using one or more binary images. For this purpose, the input image is converted into binary images using one or more threshold values and the threshold value type.

Setting a single threshold value corresponds to calling `F_VN_Threshold`. To do this, `fThresholdStep` must be set to 0. Only the value of `fMinThreshold` is used.

Alternatively, several threshold values can be taken into account. The first threshold value is `fMinThreshold`. Starting from this value, `fThresholdStep` is added for the next threshold value until the threshold value is greater than `fMaxThreshold`. If, for example, the three threshold values 100, 150 and 200 are to be taken into account, the parameterization is as follows:

```plaintext
stBlobParams.fMinThreshold := 100;
stBlobParams.fMaxThreshold := 200;
stBlobParams.fThresholdStep := 50;
```

Same contours with different threshold values

It may happen that the same contour is found at different threshold values. Identical contours are identified by means of a center point comparison. In this case `fMinBlobDistance` specifies the minimum center distance of different contours. If the distance is not reached, the contour is selected according to `eBlobCombination`.

In addition, multiple finding of a contour at different threshold values can be a filter criterion. How often a contour has to be found in order to be considered is specified via `nMinRepeatability`.

Filter

In the second step, the contours found are filtered according to the defined parameters. The individual filter criteria can be activated via a Boolean variable.

Area in pixels

- Parameter: `bFilterByArea`, `fMinArea`, `fMaxArea`
- Alternatively, the area of a contour can be determined via `F_VN_ContourArea`.

Circularity

- Parameter: `bFilterByCircularity`, `fMinCircularity`, `fMaxCircularity`
Value range from 0 to 1; a perfect circle has a circularity of 1.

Alternatively, the circularity of a contour can be determined via F_VN_ContourCircularity [856].

Convexity

Parameter: bFilterByConvexity, fMinConvexity, fMaxConvexity

The value range is from 0 to 1. A perfectly convex shape has a convexity of 1.

Eccentricity

Parameter: bFilterByEccentricity, fMinEccentricity, fMaxEccentricity

Value range from 0 to 1; a perfect circle has an eccentricity of 0.

Alternatively, the eccentricity of a contour can be determined via F_VN_ContourEccentricity [857].

Inertia ratio

Parameter: bFilterByInertiaRatio, fMinInertiaRatio, fMaxInertiaRatio

Value range from 0 to 1; shapes with the same height and width have an inertia ratio of 1.

Application

VAR
  stBlobParams : TcVnParamsBlobDetection;
END_VAR

stBlobParams.bFilterByArea := TRUE;
stBlobParams.fMinArea := 100;
stBlobParams.fMaxArea := 100_000;

hr := F_VN_DetectBlobs(  
  ipSrcImage := ipImageIn,  
  ipBlobContours := ipContours,  
  stParams := stBlobParams,  
  hrPrev := hr
);

Sample

- Blob Detection with watchdog monitoring [1390]

Related functions

- F_VN_FindContours(Exp) [1037] for general contour finding
- F_VN_FindContourHierarchyExp [1033] with return of the hierarchy
- F_VN_DetectBlobs [1029] with integrated filtering of the contours

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
### F_VN_DetectBlobsExp

Dectects blob-contents. Applies a threshold, a contour search and offers several options for filtering the found contours. Provides easy setup for multiple thresholds and combination of results. (expert function)

Can use available TwinCAT Job Tasks for executing parallel code regions.

Can return partial results when canceled by Watchdog.

#### Syntax

**Definition:**

```c
FUNCTION F_VN_DetectBlobsExp : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipBlobContours : Reference To ITcVnContainer;
    stParams : Reference To TcVnParamsBlobDetection;
    aOffset : Reference To TcVnPiont;
    hrPrev : HRESULT;
END_VAR
```

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image (USINT or REAL, 1 channel or 3 channel (3 channel input is expected to be RGB and internally converted to Gray))</td>
</tr>
<tr>
<td>ipBlobContours</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container which is filled with the found contours</td>
</tr>
<tr>
<td>stParams</td>
<td>Reference To TcVnParamsBlobDetection [211]</td>
<td>Parameters to filter the detected contours.</td>
</tr>
<tr>
<td>aOffset</td>
<td>Reference To TcVnPiont [151]</td>
<td>Offset by which every contour point is shifted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

#### Return value

**HRESULT [135]**

#### Further information

The function **F_VN_DetectBlobsExp** is the expert version of **F_VN_DetectBlobs** [1029]. It contains additional parameters.
Parameter

Input image

The input image \texttt{ipSrcImage} must be a 1-channel or 3-channel image of the type \texttt{USINT} or \texttt{REAL}. 3-channel images are interpreted as RGB and converted internally into grayscale images.

List of contours found \textit{(Return value)}

All contours found in the input image are returned in the container \texttt{ipBlobContours}.

Parameters for blob detection

The parameters \texttt{stParams} of the structure \texttt{TcVnParamsBlobDetection} define which features a contour in the input image must have in order to be found and returned.

Offset \textit{(Expert)}

The offset \texttt{aOffset} defines constant X/Y values by which all contour points are shifted. This can be helpful when contours are sought in an ROI and their positions should be referenced to the original image.

Related functions

- \texttt{F_VN_FindContours(Exp)} for general contour finding
- \texttt{F_VN_FindContourHierarchyExp} with return of the hierarchy
- \texttt{F_VN_DetectBlobs} with integrated filtering of the contours

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.13.1.4 \texttt{F_VN_FindContourHierarchyExp}

Search for object contours in a binary image and determine their hierarchical relationship. (expert function)

Can use available TwinCAT Job Tasks for executing parallel code regions.

Can return partial results when canceled by Watchdog.

Syntax

Definition:

\begin{verbatim}
FUNCTION F_VN_FindContourHierarchyExp : HRESULT
VAR_INPUT
  ipSrcImage : ITCvImage;
  ipContours : Reference To ITCvContainer;
  ipHierarchy : Reference To ITCvContainer;
  eRetrievalMode : ETCvContourRetrievalMode;
  eApproximationMethod : ETCvContourApproximationMethod;
  aOffset : Reference To TcVnPoint;
  hrPrev : HRESULT;
\end{verbatim}
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image (1 channel, binary)</td>
</tr>
<tr>
<td>ipContours</td>
<td>Reference To ITcVnContainer</td>
<td>Returns a container which is filled with the found contours</td>
</tr>
<tr>
<td>eRetrievalMode</td>
<td>ETcVnContourRetrievalMode</td>
<td>Specifies which contours are retrieved and how their relationship is encoded</td>
</tr>
<tr>
<td>eApproximationMethod</td>
<td>ETcVnContourApproximationMethod</td>
<td>Contour encoding</td>
</tr>
<tr>
<td>aOffset</td>
<td>Reference To TcVnPoint</td>
<td>Offset by which every contour point is shifted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Further information**

The function `F_VN_FindContourHierarchyExp` is an extension of the function `F_VN_FindContoursExp` [1040] by the return of the contour hierarchy.

**Parameter**

**Input image**

The input image `ipSrcImage` must have precisely 1 channel and should be a binary image.

If the input is not a binary image, it will be converted internally to one by means of a threshold value of 1. Therefore, the input image should be manually pre-processed with a threshold value or a binary segmentation.

**List of contours found** (Return value)

All contours found in the input image are returned in the container `ipContours`. The container is a two-dimensional array of points and therefore has the type `ContainerType_Vector_Vector_TcVnPoint2_DINT`.

The contours that are found are returned in `ipContours`. Each contour consists of a sum of points which are combined into a container [143]. Another higher-level container then consolidates all contours.
Hierarchy (Expert, return value)

The parameter `ipHierarchy` returns the hierarchy of the contours found in a container of the type `TcVnVector4_DINT`. The vectors of the container have the following elements:

- [0] – NEXT - Index of the next contour on the same level
- [1] – PREVIOUS - Index of the previous contour on the same level
- [3] – PARENT - Index of parent contour

Search mode (Expert)

The search mode `eRetrievalMode` of the type `ETcVnContourRetrievalMode` defines which contours are to be sought:

- TCVN_CRM_LIST: All contours
- TCVN_CRM_EXTERNAL: Only outer contours (as with `F_VN_FindContours`)
- TCVN_CRM_CONNECTED_COMPONENTS: Contours with a two-layer hierarchy
- TCVN_CRM_TREE: All contours; used in interaction with the function `F_VN_FindContourHierarchyExp` in order to determine the hierarchy of the contours
- TCVN_CRM_FLOODFILL: Returns the contours found with the help of a floodfill algorithm (available only for DINT images)

Approximation method (Expert)

The approximation method `eApproximationMethod` of the type `ETcVnContourApproximationMethod` specifies whether the contours are to be described on the basis of all surrounding points (TCVN_CAM_NONE) or whether the number of describing points are to be reduced with the help of approximation.

Offset (Expert)

The offset `aOffset` defines constant X/Y values by which all contour points are shifted. This can be helpful when contours are sought in an ROI and their positions should be referenced to the original image.

Examples of contour hierarchies

Via `eRetrievalMode`, not only is it possible to specify whether all contours or just the outer contours are to be found, but also how contours are to be sorted and how they are hierarchically related to one another. This can be read from the parameter `ipHierarchy`:

- TCVN_CRM_EXTERNAL
  Only the outer contours are returned as a sorted list. The Child [2] and Parent [3] relationships are not determined and are invalid with -1.
• **TCVN_CRM_LIST**
  
  All contours are returned as a sorted list. The Child \( [2] \) and Parent \( [3] \) relationships are not determined and are invalid with \(-1\).

```
[0] [1, -1, -1, -1]
[1] [2, 0, -1, -1]
[2] [-1, 1, -1, -1]
```

--- Hierarchy ---

```
[0] [1, -1, -1, -1]
[1] [2, 0, -1, -1]
[2] [-1, 1, -1, -1]
```

• **TCVN_CRM_CONNECTED_COMPONENTS**

  All contours are sorted according to a 2-step hierarchy. The outer contours are considered first, then in the second step the contour lying directly within them. If there are further contours within the second step, they are set to the first step like outer contours.

  In this sample, this means that the inner Child \( [2] \) contours (green in the picture) cannot be Parent \( [3] \) contours (red in the picture) at the same time. Therefore, contours that would lie in a tree structure at Level 3 are regarded again as outer contours and set to Level 1 (Parent \( [3] \) = \(-1\)), in this case contours 1 and 2.

```
[0] [1, -1, -1, -1]
[1] [2, 0, -1, -1]
[2] [3, 1, -1, -1]
[3] [4, 2, -1, -1]
[4] [5, 3, -1, -1]
[5] [6, 4, -1, -1]
[6] [-1, 4, -1, -1]
```

--- Hierarchy ---

```
[0] [1, -1, -1, -1]
[1] [2, 0, -1, -1]
[2] [3, 1, -1, -1]
[3] [4, 2, -1, -1]
[4] [5, 3, -1, -1]
[5] [6, 4, -1, -1]
[6] [-1, 4, -1, -1]
```
• TCVN_CRM_TREE

All contours are set according to a tree structure. It is possible for contours to be both Child [2] and Parent [3] contours, depending on how they are nested within one another.

--- Hierarchy ---

[1] [N, P, C, P]
[0] [1, -1, -1, -1]
[1] [6, 0, 2, -1]
[2] [-1, -1, 3, 1]
[3] [4, -1, -1, 2]
[4] [-1, 3, 5, 2]
[5] [-1, -1, -1, 4]
[6] [-1, 1, -1, -1]

Samples

• Find Contour - Hierarchy & Retrieval Mode [1401], for the parameter ETcVnContourRetrievalMode [178]
• Find Contour - Approximation Method [1398], for parameter ETcVnContourApproximationMethod [175]
• Find contour instead of Blob Detection [1396], for comparison with function F_VN_DetectBlobs [1029]

Related functions

• F_VN_FindContours(Exp) [1037] for general contour finding
• F_VN_FindContourHierarchyExp [1033] with return of the hierarchy
• F_VN_DetectBlobs [1029] with integrated filtering of the contours

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.13.1.5 F_VN_FindContours

F_VN_FindContours

iptSrcImage ITCvnImage HRESULT F_VN_FindContours
iptContours Reference To ITCvnContainer
hrPrev HRESULT

Search for object contours in a binary image. Returns only external contours. Can use available TwinCAT Job Tasks for executing parallel code regions. Can return partial results when canceled by Watchdog.
Syntax

Definition:

FUNCTION F_VN_FindContours : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  ipContours : Reference To ITcVnContainer;
  hrPrev      : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image (1 channel, binary)</td>
</tr>
<tr>
<td>ipContours</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container which is filled with the found contours (ContainerType_Vector_Vector_TcVnPoint2_DINT; The elements of this container are single contours. Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Further information

The function F_VN_FindContours detects outer contours in the input image.

Parameter

Input image

The input image ipSrcImage must have precisely 1 channel and should be a binary image.

If the input is not a binary image, it will be converted internally to one by means of a threshold value of 1. Therefore, the input image should be manually pre-processed with a threshold value or a binary segmentation.

List of contours found (Return value)

All contours found in the input image are returned in the container ipContours. The container is a two-dimensional array of points and therefore has the type ContainerType_Vector_Vector_TcVnPoint2_DINT.

The contours that are found are returned in ipContours. Each contour consists of a sum of points which are combined into a container [143]. Another higher-level container then consolidates all contours.

Expert parameters

The expert version F_VN_FindContoursExp [1040] contains additional parameters. This enables inner contours to be found too, for example.

Application

The finding and processing of contours looks like this, for example:
Pre-processing

As a binary image is to be used for the input image \( \text{ipSrcImage} \), a threshold value or binary segmentation should be applied to the image prior to the function call, e.g. using the function \( \text{F_VN_Threshold} \):

\[
\text{hr} := \text{F_VN_Threshold(ipImageIn, ipImageWork, 128, 255, TCVN_TT_BINARY, hr)};
\]

Function call

Nothing further needs to be observed with regard to the function call. Parameter settings are only necessary with the expert version \( \text{F_VN_FindContoursExp} \).

\[
\begin{align*}
\text{hr} & := \text{F_VN_FindContours}( \cr
\text{ipSrcImage} & := \text{ipImageIn}, \\
\text{ipContours} & := \text{ipContours}, \\
\text{hrPrev} & := \text{hr} 
\end{align*}
\]

Further processing

\( \text{F_VN_GetForwardIterator} \) can be used to apply an iterator to the higher-level container, which makes it possible to retrieve the containers of the individual contours one after the other via \( \text{F_VN_GetContainer} \) in order to analyze them with the Contour Analysis functions, for example. Remember to increment the iterator with \( \text{F_VN_IncrementIterator} \).

\[
\begin{align*}
\text{hr} & := \text{F_VN_GetForwardIterator(ipContours, ipIterator, hr)}; \\
\text{WHILE} & \text{SUCCEEDED(hr) AND THEN ipIterator.CheckIfEnd() <> S_OK DO} \\
\text{hr} & := \text{F_VN_GetContainer(ipIterator, ipContour, hr)}; \\
\text{hr} & := \text{F_VN_IncrementIterator(ipIterator, hr)}; \\
\end{align*}
\]

Alternatively you can also retrieve a single contour from the container of all contours with \( \text{F_VN_GetAt} \).

The number of contours found can be queried with \( \text{F_VN_GetNumberOfElements} \).

Visualization

Use the function \( \text{F_VN_DrawContours(Exp)} \) to draw the contours found.

\[
\begin{align*}
\text{hr} & := \text{F_VN_DrawContours}( \cr
\text{ipContours} & := \text{ipContours}, \\
\text{nContourIndex} & := -1, \\
\text{ipDestImage} & := \text{ipImageRes}, \\
\text{aColor} & := \text{aColorRed}, \\
\text{nThickness} & := 5, \\
\text{hrPrev} & := \text{hr} 
\end{align*}
\]

All contours in the contour list \( \text{ipContours} \) are drawn by setting \( \text{nContoursIndex} \) to \(-1\).

Samples

- Find contour instead of Blob Detection

Related functions

- \( \text{F_VN_FindContours(Exp)} \) for general contour finding
- \( \text{F_VN_FindContourHierarchyExp} \) with return of the hierarchy
- \( \text{F_VN_DetectBlobs} \) with integrated filtering of the contours

Required License

TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.13.1.6  F_VN_FindContoursExp

Search for object contours in a binary image. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```c
FUNCTION F_VN_FindContoursExp : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  ipContours : Reference To ITcVnContainer;
  eRetrievalMode : ETcVnContourRetrievalMode;
  eApproximationMethod : ETcVnContourApproximationMethod;
  aOffset : Reference To TcVnPoint;
  hrPrev : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [► 383]</td>
<td>Source image (1 channel, binary)</td>
</tr>
<tr>
<td>ipContours</td>
<td>Reference To ITcVnContainer [► 345]</td>
<td>Returns a container which is filled with the found contours (ContainerType_Vector_Vector_TcVnPoint2_DINT; The elements of this container are single contours. Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>eRetrievalMode</td>
<td>ETcVnContourRetrievalMode [► 178]</td>
<td>Specifies which contours are retrieved and how their relationship is encoded</td>
</tr>
<tr>
<td>eApproximationMethod</td>
<td>ETcVnContourApproximationMethod [► 175]</td>
<td>Contour encoding</td>
</tr>
<tr>
<td>aOffset</td>
<td>Reference To TcVnPoint [► 151]</td>
<td>Offset by which every contour point is shifted</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [► 135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [► 135]
Further information

The function \( F_{\,VN}\_FindContoursExp \) is the expert version of \( F_{\,VN}\_FindContours \). It contains additional parameters.

Parameter

Input image

The input image \( ipSrcImage \) must have precisely 1 channel and should be a binary image. If the input is not a binary image, it will be converted internally to one by means of a threshold value of 1. Therefore, the input image should be manually pre-processed with a threshold value or a binary segmentation.

List of contours found (Return value)

All contours found in the input image are returned in the container \( ipContours \). The container is a two-dimensional array of points and therefore has the type \( ContainerType\_Vector\_Vector\_TcVnPoint2\_DINT \).

The contours that are found are returned in \( ipContours \). Each contour consists of a sum of points which are combined into a container. Another higher-level container then consolidates all contours.

Search mode (Expert)

The search mode \( eRetrievalMode \) of the type \( ETcVnContourRetrievalMode \) defines which contours are to be sought:

- \( TCVN\_CRM\_LIST \): All contours
- \( TCVN\_CRM\_EXTERNAL \): Only outer contours (as with \( F_{\,VN}\_FindContours \))
- \( TCVN\_CRM\_CONNECTED\_COMPONENTS \): Contours with a two-layer hierarchy
- \( TCVN\_CRM\_TREE \): All contours; used in interaction with the function \( F_{\,VN}\_FindContourHierarchyExp \) in order to determine the hierarchy of the contours
- \( TCVN\_CRM\_FLOODFILL \): Returns the contours found with the help of a floodfill algorithm (available only for DINT images)

Approximation method (Expert)

The approximation method \( eApproximationMethod \) of the type \( ETcVnContourApproximationMethod \) specifies whether the contours are to be described on the basis of all surrounding points (\( TCVN\_CAM\_NONE \)) or whether the number of describing points are to be reduced with the help of approximation.

Offset (Expert)

The offset \( aOffset \) defines constant X/Y values by which all contour points are shifted. This can be helpful when contours are sought in an ROI and their positions should be referenced to the original image.

Application

The finding of all contours in the input image, where the contour points are not approximated, but shifted by 100 px in the x direction and by -50 px in the y direction, looks like this, for example:

```plaintext
VAR
  aOffset : TcVnPoint := [100, -50];
END_VAR

hr := F_{\,VN}\_FindContours(
  ipSrcImage := ipImageIn,
  ipContours := ipContours,
  eRetrievalMode := TCVN\_CRM\_LIST,
```
API reference

eApproximationMethod := TCVN_CAM_NONE,
aOffset := aOffset,
hrPrev := hr
);

Samples

- Find Contour - Hierarchy & Retrieval Mode [1401], for the parameter ETcVnContourRetrievalMode [178]
- Find Contour - Approximation Method [1398], for parameter ETcVnContourApproximationMethod [175]
- Find contour instead of Blob Detection [1396], for comparison with function F_VN_DetectBlobs [1029]

Related functions

- F_VN_FindContours(Exp) [1037] for general contour finding
- F_VN_FindContourHierarchyExp [1033] with return of the hierarchy
- F_VN_DetectBlobs [1029] with integrated filtering of the contours

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.13.1.7 F_VN_HoughCircles

Search for circles using the Hough transform.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

FUNCTION F_VN_HoughCircles : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  ipCircles : Reference To ITcVnContainer;
  fInvAccuRatio : LREAL;
  fMinDist : LREAL;
  hrPrev : HRESULT;
END_VAR
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvNImage</td>
<td>Source image (USINT, 1 channel, gray-level)</td>
</tr>
<tr>
<td>ipCircles</td>
<td>Reference To ITCvNContainer</td>
<td>Returns a container which is filled with the found circles</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(ContainerType_Vector_TcVnVector3_REAL; Each container element contains</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the x coordinate of the circle center [0], the y coordinate of the circle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>center [1], and the radius [2], respectively. Non-zero interface pointers</td>
</tr>
<tr>
<td></td>
<td></td>
<td>are reused.)</td>
</tr>
<tr>
<td>fInvAccuRatio</td>
<td>LREAL</td>
<td>Inverted ratio of the accumulator size in relation to the source image's</td>
</tr>
<tr>
<td></td>
<td></td>
<td>size (must be &gt; 0. A value of 2 means that the size is halved in both</td>
</tr>
<tr>
<td></td>
<td></td>
<td>directions.)</td>
</tr>
<tr>
<td>fMinDist</td>
<td>LREAL</td>
<td>Smallest allowed distance of two circles (must be &gt; 0)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.13.1.8  F_VN_HoughCirclesExp

Search for circles using the Hough transform. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

FUNCTION F_VN_HoughCirclesExp : HRESULT
VAR_INPUT
  ipSrcImage : ITCvNImage;
  ipCircles : Reference To ITCvNContainer;
  fInvAccuRatio : LREAL;
  fMinDist : LREAL;
  nMinRadius : UDINT;
  nMaxRadius : UDINT;
  hrPrev : HRESULT;
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image (USINT, 1 channel, gray-level)</td>
</tr>
<tr>
<td>ipCircles</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container which is filled with the found circles</td>
</tr>
<tr>
<td>fInvAccuRatio</td>
<td>LREAL</td>
<td>Inverted ratio of the accumulator size in relation to the source image's size (must be &gt; 0. A value of 2 means that the size is halved in both directions.)</td>
</tr>
<tr>
<td>fMinDist</td>
<td>LREAL</td>
<td>Smallest allowed distance of two circles (must be &gt; 0)</td>
</tr>
<tr>
<td>eHoughMethod</td>
<td>ETcVnHoughMethod [191]</td>
<td>Hough method to use (GRADIENT or GRADIENT_ALT)</td>
</tr>
<tr>
<td>fParam1</td>
<td>LREAL</td>
<td>First method specific parameter (GRADIENT, GRADIENT_ALT: upper threshold for canny edge detection, which must be &gt; 0)</td>
</tr>
<tr>
<td>fParam2</td>
<td>LREAL</td>
<td>Second method specific parameter (GRADIENT: accumulator threshold for detecting circle centers, which must be &gt; 0. GRADIENT_ALT: required circle perfectness, which must be &gt; 0 and &lt; 1, 1 would be a perfect circle)</td>
</tr>
<tr>
<td>nMinRadius</td>
<td>UDINT</td>
<td>Minimum circle radius allowed</td>
</tr>
<tr>
<td>nMaxRadius</td>
<td>UDINT</td>
<td>Maximum circle radius allowed (if 0, the value is internally set to max(image rows, image columns))</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.13.1.9 F_VN_HoughLines

F_VN_HoughLines

ipSrcImage ITcVnImage HRESULT F_VN_HoughLines
ipLines Reference To ITcVnContainer
fDistRes LREAL
fAngleRes LREAL
nAccuThreshold UDINT
hrPrev HRESULT

Search for lines using the standard Hough transform.

Syntax

Definition:
FUNCTION F_VN_HoughLines : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  ipLines : Reference To ITcVnContainer;
  fDistRes : LREAL;
  fAngleRes : LREAL;
  nAccuThreshold : UDINT;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image (USINT, 1 channel, binary, may be modified by the function)</td>
</tr>
<tr>
<td>ipLines</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container which is filled with the found lines (ContainerType_Vector_TcVnVector2_REAL; Each container element contains the distance from the origin [0] in pixels and the rotation angle [1] in radians, respectively. Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>fDistRes</td>
<td>LREAL</td>
<td>Distance resolution of the accumulator (in pixels, must be &gt; 0)</td>
</tr>
<tr>
<td>fAngleRes</td>
<td>LREAL</td>
<td>Angle resolution of the accumulator (in radians, must be &gt; 0)</td>
</tr>
<tr>
<td>nAccuThreshold</td>
<td>UDINT</td>
<td>Accumulator threshold</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
### 6.4.13.1.10 F_VN_HoughLinesExp

**F_VN_HoughLinesExp**

```plaintext
FUNCTION F_VN_HoughLinesExp : HRESULT
VAR_INPUT
    ipSrcImage : ITCvImage;
    ipLines : Reference To ITCvnContainer;
    fDistRes : LREAL;
    fAngleRes : LREAL;
    nAccuThreshold : UDINT;
    fDistResDiv : LREAL;
    fAngleResDiv : LREAL;
    hrPrev : HRESULT;
END_VAR
```

Search for lines using the standard Hough transform. (expert function)

**Syntax**

**Definition:**

Search for lines using the standard Hough transform. (expert function)

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvImage [383]</td>
<td>Source image (USINT, 1 channel, binary, may be modified by the function)</td>
</tr>
<tr>
<td>ipLines</td>
<td>Reference To ITCvnContainer [345]</td>
<td>Returns a container which is filled with the found lines (ContainerType_Vector_TcVnVector2_REAL; Each container element contains the distance from the origin [0] in pixels and the rotation angle [1] in radians, respectively. Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>fDistRes</td>
<td>LREAL</td>
<td>Distance resolution of the accumulator (in pixels, must be &gt; 0)</td>
</tr>
<tr>
<td>fAngleRes</td>
<td>LREAL</td>
<td>Angle resolution of the accumulator (in radians, must be &gt; 0)</td>
</tr>
<tr>
<td>nAccuThreshold</td>
<td>UDINT</td>
<td>Accumulator threshold</td>
</tr>
<tr>
<td>fDistResDiv</td>
<td>LREAL</td>
<td>Divisor of the distance resolution for the multi-scale Hough transform (&gt;= 0, must be &gt; 0 if fAngleResDiv is &gt; 0)</td>
</tr>
<tr>
<td>fAngleResDiv</td>
<td>LREAL</td>
<td>Divisor of the angle resolution for the multi-scale Hough transform (&gt;= 0, must be &gt; 0 if fDistResDiv is &gt; 0)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.13.1.11  F_VN_HoughLinesP

```
FUNCTION F_VN_HoughLinesP : HRESULT
VAR_INPUT
    ipSrcImage     : ITcVnImage;
    ipLines        : Reference To ITcVnContainer;
    fDistRes       : LREAL;
    fAngleRes      : LREAL;
    nAccuThreshold : UDINT;
    hrPrev         : HRESULT;
END_VAR
```

Search for line segments using the probabilistic Hough transform.

Syntax

**Definition:**

```
FUNCTION F_VN_HoughLinesP : HRESULT
VAR_INPUT
    ipSrcImage     : ITcVnImage;
    ipLines        : Reference To ITcVnContainer;
    fDistRes       : LREAL;
    fAngleRes      : LREAL;
    nAccuThreshold : UDINT;
    hrPrev         : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage[383]</td>
<td>Source image (USINT, 1 channel, binary, may be modified by the function)</td>
</tr>
<tr>
<td>ipLines</td>
<td>Reference To ITcVnContainer[345]</td>
<td>Returns a container which is filled with the found line segments (ContainerType_Vector_TcVnVector4_DINT; Each container element contains the two ending points. Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>fDistRes</td>
<td>LREAL</td>
<td>Distance resolution of the accumulator (in pixels, must be &gt; 0)</td>
</tr>
<tr>
<td>fAngleRes</td>
<td>LREAL</td>
<td>Angle resolution of the accumulator (in radians, must be &gt; 0)</td>
</tr>
<tr>
<td>nAccuThreshold</td>
<td>UDINT</td>
<td>Accumulator threshold</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.13.1.12  F_VN_HoughLinesPExp

Search for line segments using the probabilistic Hough transform. (expert function)

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_HoughLinesPExp : HRESULT
VAR_INPUT
    ipSrcImage : ITCvImage;
    ipLines   : Reference To ITCvContainer;
    fDistRes  : LREAL;
    fAngleRes : LREAL;
    nAccuThreshold : UDINT;
    fMinLineLength : LREAL;
    fMaxLineGap : LREAL;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvImage [383]</td>
<td>Source image (USINT, 1 channel, binary, may be modified by the function)</td>
</tr>
<tr>
<td>ipLines</td>
<td>Reference To ITCvContainer [345]</td>
<td>Returns a container which is filled with the found line segments (ContainerType_Vector_TcVnVector4_DINT; Each container element contains the two ending points. Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>fDistRes</td>
<td>LREAL</td>
<td>Distance resolution of the accumulator (in pixels, must be &gt; 0)</td>
</tr>
<tr>
<td>fAngleRes</td>
<td>LREAL</td>
<td>Angle resolution of the accumulator (in radians, must be &gt; 0)</td>
</tr>
<tr>
<td>nAccuThreshold</td>
<td>UDINT</td>
<td>Accumulator threshold</td>
</tr>
<tr>
<td>fMinLineLength</td>
<td>LREAL</td>
<td>Minimum line length to search for</td>
</tr>
<tr>
<td>fMaxLineGap</td>
<td>LREAL</td>
<td>Maximum gap between points on the same line</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]
**Required License**
TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.13.1.13 F_VN_MatchImageHuMoments**

Compare two images using the Hu moment invariants.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_MatchImageHuMoments : HRESULT
VAR_INPUT
   ipImage1 : ITcVnImage;
   ipImage2 : ITcVnImage;
   eComparisonMethod : ETcVnContoursMatchComparisonMethod;
   fDissimilarity : Reference To LREAL;
   hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage1</td>
<td>ITcVnImage</td>
<td>First image (1 channel)</td>
</tr>
<tr>
<td>ipImage2</td>
<td>ITcVnImage</td>
<td>Second image (1 channel)</td>
</tr>
<tr>
<td>eComparisonMethod</td>
<td>ETcVnContoursMatchComparisonMethod</td>
<td>Method used for comparing the Hu moment invariants of the images</td>
</tr>
<tr>
<td>fDissimilarity</td>
<td>Reference To LREAL</td>
<td>Returns the dissimilarity of the image Hu moment invariants depending on the chosen comparison method</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**
TC3 Vision Matching
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.13.1.14 F_VN_MatchTemplate

Match a template image with every location in the source image (using the TCVN_TMM_CCORR_NORMED method) and save the comparison results. Can return partial results when canceled by Watchdog.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_MatchTemplate : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipTemplateImage : ITcVnImage;
    ipResultImage : Reference To ITcVnImage;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image (USINT or REAL, 1 or 3 channels)</td>
</tr>
<tr>
<td>ipTemplateImage</td>
<td>ITcVnImage</td>
<td>Template image (same type as ipSrcImage, smaller width and height)</td>
</tr>
<tr>
<td>ipResultImage</td>
<td>Reference To ITcVnImage</td>
<td>Returns the result image (REAL, normalized to [0..1], 1 channel, dimensions: (ipSrcImage.width - ipTemplateImage.width + 1) x (ipSrcImage.height - ipTemplateImage.height + 1). The best match is the global maximum. The position in ipResultImage is the top-left corner of ipTemplateImage position in ipSrcImage)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Matching

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.13.1.15  F_VN_MatchTemplateAndEvaluate

**Match a template image with every location in the source image (using the TCVN_TMM_CCORR_NORMED method) and evaluate the comparison results. Returns a sorted list of possible matches (best match first). Can use available TwinCAT Job Tasks for executing parallel code regions. Can return partial results when canceled by Watchdog.**

**Syntax**

**Definition:**

FUNCTION F_VN_MatchTemplateAndEvaluate : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  ipTemplateImage : ITcVnImage;
  ipMatches : Reference To ITcVnContainer;
  fMatchThreshold : REAL;
  hrPrev : HRESULT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image (USINT or REAL, 1 or 3 channels)</td>
</tr>
<tr>
<td>ipTemplateImage</td>
<td>ITcVnImage [383]</td>
<td>Template image (same type as ipSrcImage, smaller width and height)</td>
</tr>
<tr>
<td>ipMatches</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns the matching positions (ContainerType_Vector_TcVnPoint2_DINT, where each element represents the top-left corner of ipTemplateImage) in ipSrcImage, sorted by relevance (best match first)</td>
</tr>
<tr>
<td>fMatchThreshold</td>
<td>REAL</td>
<td>Threshold to separate relevant from irrelevant matches (0..1, 1.0 would be a perfect match. To find a suitable value, you could evaluate some sample result images of F_VN_MatchTemplate.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Matching

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.13.1.16  F_VN_MatchTemplateAndEvaluateExp

Match a template image with every location in the source image and evaluate the comparison results. Returns a sorted list of possible matches (best match first). (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

FUNCTION F_VN_MatchTemplateAndEvaluateExp : HRESULT
VAR_INPUT
  ipSrcImage : ITCvNImage;
  ipTemplateImage : ITCvNImage;
  ipMatches : Reference To ITCvNContainer;
  fMatchThreshold : REAL;
  eMatchMethod : ETCvNTemplateMatchMethod;
  ipTemplateMask : ITCvNImage;
  fScaleFactor : REAL;
  eInterpolationType : ETCvNInterpolationType;
  ipMatchValues : Reference To ITCvNContainer;
  hr Prev : HRESULT;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvNImage [383]</td>
<td>Source image (USINT or REAL, 1 or 3 channels)</td>
</tr>
<tr>
<td>ipTemplateImage</td>
<td>ITCvNImage [383]</td>
<td>Template image (same type as ipSrcImage, smaller width and height)</td>
</tr>
<tr>
<td>ipMatches</td>
<td>Reference To ITCvNContainer [345]</td>
<td>Returns the matching positions (ContainerType_Vector_TcVnPoint2_DINT, where each element represents the top-left corner of ipTemplateImage) in ipSrcImage, sorted by relevance (best match first)</td>
</tr>
<tr>
<td>fMatchThreshold</td>
<td>REAL</td>
<td>Threshold to separate relevant from irrelevant matches (0..1 for NORMED methods, otherwise dependent on template size and content. To find a suitable value, you could evaluate some sample result images of F_VN_MatchTemplateExp.)</td>
</tr>
<tr>
<td>eMatchMethod</td>
<td>ETCvNTemplateMatchMethod [200]</td>
<td>Specifies the template match method</td>
</tr>
<tr>
<td>ipTemplateMask</td>
<td>ITCvNImage [383]</td>
<td>Optional mask for ipTemplateImage (only supported for SQDIFF and CCORR_NORMED, same type and size as ipTemplateImage)</td>
</tr>
<tr>
<td>fScaleFactor</td>
<td>REAL</td>
<td>Factor (0..1] to reduce source and template image width and height for better performance (but less accuracy!)</td>
</tr>
<tr>
<td>eInterpolationType</td>
<td>ETCvNInterpolationType [192]</td>
<td>Image resize interpolation type (only used if fScaleFactor != 1, TCVN_IT_BILINEAR recommended for most cases)</td>
</tr>
<tr>
<td>ipMatchValues</td>
<td>Reference To ITCvNContainer [345]</td>
<td>Optionally returns the matching values (ContainerType_Vector_REAL, same size and sort order as ipMatches. Set to 0 if not required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Matching

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.13.1.17 F_VN_MatchTemplateExp

```plaintext
F_VN_MatchTemplateExp

ipSrcImage ITCvNImage
ipTemplateImage ITCvNImage
ipResultImage Reference To ITCvNImage
eMatchMethod ETCvNTemplateMatchMethod
ipTemplateMask ITCvNImage
hrPrev HRESULT
```

Translated F_VN_MatchTemplateExp:

```plaintext
F_VN_MatchTemplateExp

ipSrcImage ITCvNImage
ipTemplateImage ITCvNImage
ipResultImage Reference To ITCvNImage
eMatchMethod ETCvNTemplateMatchMethod
ipTemplateMask ITCvNImage
hrPrev HRESULT
```
Match a template image with every location in the source image and save the comparison results. (expert function)
Can return partial results when canceled by Watchdog.

**Syntax**

**Definition:**

\[
\text{FUNCTION F_VN_MatchTemplateExp : HRESULT }
\]
\[
\text{VAR_INPUT }
\]
\[
\begin{align*}
\hspace{0.5cm} \text{ipSrcImage} & : \text{ITcVnImage}; \\
\hspace{0.5cm} \text{ipTemplateImage} & : \text{ITcVnImage}; \\
\hspace{0.5cm} \text{ipResultImage} & : \text{Reference To ITcVnImage}; \\
\hspace{0.5cm} \text{eMatchMethod} & : \text{ETcVnTemplateMatchMethod}; \\
\hspace{0.5cm} \text{ipTemplateMask} & : \text{ITcVnImage}; \\
\hspace{0.5cm} \text{hrPrev} & : \text{HRESULT};
\end{align*}
\]
\[
\text{END_VAR}
\]

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image (USINT or REAL, 1 or 3 channels)</td>
</tr>
<tr>
<td>ipTemplateImage</td>
<td>ITcVnImage</td>
<td>Template image (same type as ipSrcImage, smaller width and height)</td>
</tr>
<tr>
<td>ipResultImage</td>
<td>Reference To ITcVnImage</td>
<td>Returns the result image (REAL, 1 channel, dimensions: (ipSrcImage.width - ipTemplateImage.width + 1) x (ipSrcImage.height - ipTemplateImage.height + 1). The best match is the global minimum (SQDIFF(_NORMED)) or maximum (CCORR(_NORMED), CCOEFF(_NORMED)). The position in ipResultImage is the top-left corner of ipTemplateImage position in ipSrcImage)</td>
</tr>
<tr>
<td>eMatchMethod</td>
<td>ETcVnTemplateMatchMethod</td>
<td>Specifies the template match method</td>
</tr>
<tr>
<td>ipTemplateMask</td>
<td>ITcVnImage</td>
<td>Optional mask for ipTemplateImage (only supported for SQDIFF and CCORR_NORMED, same type and size as ipTemplateImage)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT \(\rightarrow 135\)

**Required License**

TC3 Vision Matching

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
Find edges using the Canny edge detection algorithm. Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```c
FUNCTION F_VN_CannyEdgeDetection : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    fThresholdLow : LREAL;
    fThresholdHigh : LREAL;
    hrPrev : HRESULT;
END_VAR
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvImage</td>
<td>Source image (elements of type USINT)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITCvImage</td>
<td>Destination image (elements of type USINT. An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>fThresholdLow</td>
<td>LREAL</td>
<td>Low threshold</td>
</tr>
<tr>
<td>fThresholdHigh</td>
<td>LREAL</td>
<td>High threshold</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Further information

This function finds edges in an image and illustrates them by white lines on a black background. In addition, this expert version F_VN_CannyEdgeDetectionExp [1057] exists with advanced parameterization options.

Algorithm

The Canny Edge edge detection is a multi-stage algorithm for the most robust edge detection possible. It consists of the following steps:

1. Reduction of noise by means of Gaussian filter.
2. Edge finding by means of Sobel operator.
3. Line thinning by means of non-maximum suppression, i.e. the resulting lines are 1 px thick.
4. Hysteresis, so that only thick lines remain (see Threshold parameters below).
Parameter

Original image
A 1-channel 8-bit intensity image (pixel format \texttt{TCVN\_ET\_USINT}) is expected for the original image \texttt{ipSrcImage}. You can therefore use a grayscale image or the individual channels of a color image.

Result image
The result image \texttt{ipDestImage} has the same format.

Threshold values
The two thresholds \texttt{fThresholdLow} and \texttt{fThresholdHigh} determine which of the edges found are accepted: Pixels with a gradient higher than \texttt{fThresholdHigh} are accepted. Pixels with a color gradient below \texttt{fThresholdLow} are discarded. Pixels with gradients in between are only accepted if they are linked to already accepted pixels.

The two thresholds make a flexible setting possible. A hard threshold is also realizable by setting both values the same.

Important edge information may be lost if you set the thresholds too high. If the thresholds are too low, noise and other irrelevant information might be interpreted as edges.

Application

```c
hr := F\_VN\_CannyEdgeDetection(
    ipSrcImage := ipImageIn,
    ipDestImage := ipImageWork,
    fThresholdLow := 0,
    fThresholdHigh := 200,
    hr
);
```

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.13.3  F_VN_CannyEdgeDetectionExp

Find edges using the Canny edge detection algorithm. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

FUNCTION F_VN_CannyEdgeDetectionExp : HRESULT
VAR_INPUT
 ipSrcImage : ITcVnImage;
 ipDestImage : Reference To ITcVnImage;
 fThresholdLow : LREAL;
 fThresholdHigh : LREAL;
 nApertureSize : UDINT;
 bL2Gradient : BOOL;
 hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image (elements of type USINT)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (elements of type USINT. An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>fThresholdLow</td>
<td>LREAL</td>
<td>Low threshold</td>
</tr>
<tr>
<td>fThresholdHigh</td>
<td>LREAL</td>
<td>High threshold</td>
</tr>
<tr>
<td>nApertureSize</td>
<td>UDINT</td>
<td>Aperture size for the Sobel operator (3, 5, 7)</td>
</tr>
<tr>
<td>bL2Gradient</td>
<td>BOOL</td>
<td>If true, the more accurate (and slower) L2 norm is used instead of the L1 norm</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Further information

This function is the expert version of F_VN_CannyEdgeDetection [1055]. It contains advanced parameterization options.

Parameter

The following parameters are additionally available in this expert version:

Aperture size

Only odd values of 3 or more are valid for nApertureSize.
**L1/L2 standard**

Two standards are available for the calculation of the gradients. The L1 standard is faster. The L2 standard, conversely, is more accurate.

\[
\begin{align*}
L_1 &= \left| \frac{dl}{dx} \right| + \left| \frac{dl}{dy} \right| \\
L_2 &= \sqrt{\left( \frac{dl}{dx} \right)^2 + \left( \frac{dl}{dy} \right)^2}
\end{align*}
\]

Set the parameter \texttt{bL2Gradient} to \texttt{FALSE} to use the L1 standard and to \texttt{TRUE} to use the L2 standard.

**Application**

An application of this function can look like this, for example:

```
hr := F_VN_CannyEdgeDetectionExp(
    ipSrcImage := ipImageIn,
    ipDestImage := ipImageWork,
    fThresholdLow := 0,
    fThresholdHigh := 200,
    nApertureSize := 5,
    bL2Gradient := TRUE,
    hr
);
```

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.13.4 F_VN_ConnectedComponents**

Computes the connected components of an image.

**Syntax**

**Definition:**

```pascal
FUNCTION F_VN_ConnectedComponents : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipLabelImage : Reference To ITcVnImage;
    nLabels : Reference To DINT;
    hrPrev : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage[383]</td>
<td>Source image with threshold applied (1 channel, USINT)</td>
</tr>
<tr>
<td>ipLabelImage</td>
<td>Reference To ITcVnImage[383]</td>
<td>Returns the labels for each source image pixel (1 channel, DINT. An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>nLabels</td>
<td>Reference To DINT</td>
<td>Returns the number of labels</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.13.5 F_VN_ConnectedComponentsExp

Computes the connected components of an image. (expert function)

#### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_ConnectedComponentsExp : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  ipLabelImage : Reference To ITcVnImage;
  nLabels : Reference To DINT;
  eConnectivity : ETcVnPixelConnectivity;
  eLabelType : ETcVnElementType;
  eAlgorithm : ETcVnConnectedComponentsAlgorithm;
  hrPrev : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image with threshold applied (1 channel, USINT)</td>
</tr>
<tr>
<td>ipLabelImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Returns the labels for each source image pixel (1 channel, type depends on eLabelType. An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>nLabels</td>
<td>Reference To DINT</td>
<td>Returns the number of labels</td>
</tr>
<tr>
<td>eConnectivity</td>
<td>ETcVnPixelConnectivity [196]</td>
<td>Selects if 4- or 8-way pixel connectivity should be used</td>
</tr>
<tr>
<td>eLabelType</td>
<td>ETcVnElementType [186]</td>
<td>Selects the type of ipLabelImage (only UINT or DINT supported)</td>
</tr>
<tr>
<td>eAlgorithm</td>
<td>ETcVnConnectedComponentsAlgorithm [174]</td>
<td>Selects the applied algorithm</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.13.6 F_VN_ConnectedComponentsWithStats

Computes the connected components and corresponding statistics of an image.

#### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_ConnectedComponentsWithStats : HRESULT
VAR_INPUT
   ipSrcImage : ITcVnImage;
   ipLabelImage : Reference To ITcVnImage;
   ipBoundingBoxes : Reference To ITcVnContainer;
   ipNumPixels : Reference To ITcVnContainer;
   ipCentroids : Reference To ITcVnContainer;
   nLabels : Reference To DINT;
   hrPrev : HRESULT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvImage [383]</td>
<td>Source image with threshold applied (1 channel, USINT)</td>
</tr>
<tr>
<td>ipLabellImage</td>
<td>Reference To ITCvImage [383]</td>
<td>Returns the labels for each source image pixel (1 channel, DINT. An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>ipBoundingBoxes</td>
<td>Reference To ITCvImageContainer [345]</td>
<td>Returns the bounding boxes for each labeled region. (ContainerType_Vector_TcVnRectangle_DINT; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>ipNumPixels</td>
<td>Reference To ITCvImageContainer [345]</td>
<td>Returns the number of pixels for each labeled region. (ContainerType_Vector_DINT; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>ipCentroids</td>
<td>Reference To ITCvImageContainer [345]</td>
<td>Returns the centroids for each labeled region. (ContainerType_Vector_TcVnPoint2_LREAL; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>nLabels</td>
<td>Reference To DINT</td>
<td>Returns the number of labels</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024,17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.13.7 F_VN_ConnectedComponentsWithStatsExp**

Computes the connected components and corresponding statistics of an image. (expert function)

**Syntax**

**Definition:**

FUNCTION F_VN_ConnectedComponentsWithStatsExp : HRESULT

VAR_INPUT

  ipSrcImage : ITCvImage;
  ipLabellImage : Reference To ITCvImage;
  ipBoundingBoxes : Reference To ITCvImageContainer;
  ipNumPixels : Reference To ITCvImageContainer;
  ipCentroids : Reference To ITCvImageContainer;
  nLabels : Reference To DINT;
  hrPrev : HRESULT;
  eConnectivity : ETCvPixelConnectivity;
  eLabelType : ETCvElementType;
  eAlgorithm : ETCvConnectedComponentsAlgorithm;
nLabels : Reference To DINT;
eConnectivity : ETcVnPixelConnectivity;
eLabelType : ETcVnElementType;
eAlgorithm : ETcVnConnectedComponentsAlgorithm;
hrPrev : HRESULT;
END_VAR

## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [138]</td>
<td>Source image with threshold applied (1 channel, USINT)</td>
</tr>
<tr>
<td>ipLabelImage</td>
<td>Reference To ITcVnImage [138]</td>
<td>Returns the labels for each source image pixel (1 channel, DINT. An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>ipBoundingBoxes</td>
<td>Reference To ITcVnContainer [145]</td>
<td>Returns the bounding boxes for each labeled region. (ContainerType_Vector_TcVnRectangle_DINT; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>ipNumPixels</td>
<td>Reference To ITcVnContainer [145]</td>
<td>Returns the number of pixels for each labeled region. (ContainerType_Vector_DINT; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>ipCentroids</td>
<td>Reference To ITcVnContainer [145]</td>
<td>Returns the centroids for each labeled region. (ContainerType_Vector_TcVnPoint2_LREAL; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>nLabels</td>
<td>Reference To DINT</td>
<td>Returns the number of labels</td>
</tr>
<tr>
<td>eConnectivity</td>
<td>ETcVnPixelConnectivity [196]</td>
<td>Selects if 4- or 8-way pixel connectivity should be used</td>
</tr>
<tr>
<td>eLabelType</td>
<td>ETcVnElementType [186]</td>
<td>Selects the type of ipLabelImage (only UINT or DINT supported)</td>
</tr>
<tr>
<td>eAlgorithm</td>
<td>ETcVnConnectedComponentsAlgorithm [174]</td>
<td>Selects the applied algorithm</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.13.8 F_VN_CountNonZeroPixels

F_VN_CountNonZeroPixels

<table>
<thead>
<tr>
<th>ipSrcImage</th>
<th>ITcVnImage</th>
<th>hrPrev</th>
<th>HRESULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>nNonZero</td>
<td>Reference To ULINT</td>
<td>F_VN_CountNonZeroPixels</td>
<td></td>
</tr>
</tbody>
</table>

Counts the non-zero pixels in a single-channel image (e.g. useful to analyze threshold results).
Syntax

Definition:

```
FUNCTION F_VN_CountNonZeroPixels : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    nNonZero   : Reference To ULINT;
    hrPrev     : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image (1 channel)</td>
</tr>
<tr>
<td>nNonZero</td>
<td>Reference To ULINT</td>
<td>Returns the number of non-zero pixels in ipSrcImage</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.13.9  F_VN_DistanceTransformation

```
F_VN_DistanceTransformation
ipSrcImage ITcVnImage
ipDestImage Reference To ITcVnImage
eDistanceType ETcVnDistanceType
eMaskSize ETcVnDistanceTransformationMask
hrPrev HRESULT
```

Calculates the distance transformation, which is the distance to the closest zero pixel in a binary image.

Syntax

Definition:

```
FUNCTION F_VN_DistanceTransformation : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    eDistanceType : ETcVnDistanceType;
    eMaskSize : ETcVnDistanceTransformationMask;
    hrPrev : HRESULT;
END_VAR
```
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image (USINT, 1 channel)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (REAL, 1 channel. An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>eDistanceType</td>
<td>ETcVnDistanceType [183]</td>
<td>Distance computation method (supported: L1, L2, C)</td>
</tr>
<tr>
<td>eMaskSize</td>
<td>ETcVnDistanceTransformationMask [182]</td>
<td>Size of the distance transformation mask</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Further information

This function performs the distance transformation of an image. In addition, there is an expert version F_VN_DistanceTransformationExp [1065] with advanced parameterization options.

### Parameter

The following parameters are available in the basic version:

#### Original image

A 1-channel 8-bit image (element type TCVN_ET_USINT) is expected for the original image `ipSrcImage`. Other image formats must first be converted.

#### Result image

The result image `ipDestImage` is returned with the element type TCVN_ET_REAL. The value of each pixel describes the distance between the respective pixel and the nearest 0-value pixel in the original image.

#### Display in the ADS Image Watch

In order to be able to display the result image in the ADS Image Watch, it must first be converted into a compatible image format. Depending on the use case, it may be useful to scale the pixel values.

#### Distance type

The distance type `eDistanceType` determines the equation with which the distance between a pixel and the nearest 0-value pixel is calculated. The following distance types from the enum ETcVnDistanceType [183] are supported:

- TCVN_DT_L1
- TCVN_DT_L2
- TCVN_DT_C

#### Mask sizes

The mask size `eMaskSize` describes the approximation of the Euclidean distance if the L2 standard is selected as the distance type.

- TCVN_DTM_PRECISE: The Euclidean distance is calculated precisely.
• TCVN_DTM_3: The Euclidean distance is approximated as the sum of distance elements within a 3x3 mask (horizontal, vertical and diagonal).
• TCVN_DTM_5: The Euclidean distance is approximated as the sum of distance elements within a 5x5 mask (horizontal, vertical, diagonal and knight's move (as in chess)).

The mask size has no effect with the distance types L1 and C. Instead, a 3x3 mask is always used internally.

Application

An application of this function can look like this, for example:

```c
// convert to binary image
hr := F_VN_Threshold(ipImageIn, ipImageWork, 128, 255, TCVN_TT_BINARY, hr);
hr := F_VN_DistanceTransformation(
    ipSrcImage := ipImageWork,
    ipDestImage := ipImageWork,
    eDistanceType := TCVN_DT_C,
    eMaskSize := TCVN_DTM_3,
    hr
);
// convert image to display it in ADS Image Watch
hr := F_VN_ConvertElementType(ipImageWork, ipImageWork, TCVN_ET_USINT, hr);
```

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.13.10 F_VN_DistanceTransformationExp

Calculates the distance transformation, which is the distance to the closest zero pixel in a binary image. Additionally, an image with component labels is created (discrete Voronoi diagram). (expert function)

Syntax

Definition:
FUNCTION F_VN_DistanceTransformationExp : HRESULT
VAR_INPUT
  ipSrcImage : ITCvImage;
  ipDestImage : Reference To ITCvImage;
  eDistanceType : ETcVnDistanceType;
  eMaskSize : ETcVnDistanceTransformationMask;
  ipDestLabels : Reference To ITCvImage;
  eLabelType : ETcVnDistanceTransformationLabel;
  hrPrev : HRESULT;
END_VAR

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvImage [383]</td>
<td>Source image (USINT, 1 channel)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITCvImage [383]</td>
<td>Destination image (REAL, 1 channel. An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>eDistanceType</td>
<td>ETcVnDistanceType [183]</td>
<td>Distance computation method (supported: L1, L2, C)</td>
</tr>
<tr>
<td>eMaskSize</td>
<td>ETcVnDistanceTransformationMask [182]</td>
<td>Size of the distance transformation mask (PRECISE not supported for label computation)</td>
</tr>
<tr>
<td>ipDestLabels</td>
<td>Reference To ITCvImage [383]</td>
<td>Returns the component labels (Discrete Voronoi diagram; DINT, 1 channel. An appropriate image will be created if required.)</td>
</tr>
<tr>
<td>eLabelType</td>
<td>ETcVnDistanceTransformationLabel [182]</td>
<td>Type of the labels</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.13.11 F_VN_ImageAverage

F_VN_ImageAverage

- **F_VN_ImageAverage**
- **ipSrcImage** ITCvImage
- **aAverage** Reference To TcVnVector4_LREAL
- **hrPrev** HRESULT

Computes the (channel-wise) average pixel value of an image.

### Syntax

Definition:
FUNCTION F_VN_ImageAverage : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
aAverage   : Reference To TcVnVector4_LREAL;
hrPrev     : HRESULT;
END_VAR

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image (1 - 4 channels)</td>
</tr>
<tr>
<td>aAverage</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Returns the (channel-wise) average pixel value of ipSrcImage</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.13.12 F_VN_ImageAverageExp

**F_VN_ImageAverageExp**

- **ipSrcImage** `ITcVnImage`
- **aAverage** `Reference To TcVnVector4_LREAL`
- **ipMask** `ITcVnImage`
- **hrPrev** `HRESULT`

Computes the (channel-wise) average pixel value of an image. (expert function)

### Syntax

**Definition:**

FUNCTION F_VN_ImageAverageExp : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
aAverage : Reference To TcVnVector4_LREAL;
ipMask : ITcVnImage;
hrPrev : HRESULT;
END_VAR
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [1383]</td>
<td>Source image (1 - 4 channels)</td>
</tr>
<tr>
<td>aAverage</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Returns the (channel-wise) average pixel value of ipSrcImage</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITcVnImage [383]</td>
<td>Mask of type USINT (1 channel)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.13.13 **F_VN_ImageAverageStdDev**

Computes the (channel-wise) average pixel value and the corresponding standard deviation of an image.

**Syntax**

**Definition:**

FUNCTION F_VN_ImageAverageStdDev : HRESULT  
VAR_INPUT  
  ipSrcImage : ITcVnImage;  
  aAverage : Reference To TcVnVector4_LREAL;  
  aStdDev : Reference To TcVnVector4_LREAL;  
  hrPrev : HRESULT;  
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image (1 - 4 channels)</td>
</tr>
<tr>
<td>aAverage</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Returns the (channel-wise) average pixel value of ipSrcImage</td>
</tr>
<tr>
<td>aStdDev</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Returns the (channel-wise) pixel value standard deviation of ipSrcImage</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.13.14 F_VN_ImageAverageStdDevExp

Computes the (channel-wise) average pixel value and the corresponding standard deviation of an image. (expert function)

**Syntax**

**Definition:**

```c
FUNCTION F_VN_ImageAverageStdDevExp : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    aAverage : Reference To TcVnVector4_LREAL;
    aStdDev : Reference To TcVnVector4_LREAL;
    ipMask : ITcVnImage;
    hrPrev : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [1383]</td>
<td>Source image (1 - 4 channels)</td>
</tr>
<tr>
<td>aAverage</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Returns the (channel-wise) average pixel value of ipSrcImage</td>
</tr>
<tr>
<td>aStdDev</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Returns the (channel-wise) pixel value standard deviation of ipSrcImage</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITcVnImage [1383]</td>
<td>Mask of type USINT (1 channel)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.13.15 F_VN_Ima g eCenterOfMass

Computes the center of mass of an image.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_ImageCenterOfMass : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  aCenterOfMass : Reference To TcVnPoint2_LREAL;
  hrPrev : HRESULT;
END_VAR
```

```c
ipSrcImage
F_VN_ImageCenterOfMass
HRESULT
F_VN_ImageCenterOfMass
hrPrev
HRESULT
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [1383]</td>
<td>Source image (1 channel)</td>
</tr>
<tr>
<td>aCenterOfMas s</td>
<td>Reference To TcVnPoint2_LREAL [151]</td>
<td>Returns the center of mass of the image</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Further information

This function calculates the center of mass of the input image. The expert version F_VN_ImageCenterOfMassExp [1071] exists with advanced parameterization options.

Parameter

Input image

The input image ipSrcImage must be a 1-channel image. If you wish to calculate the center of mass of a multi-channel image, you have the following options:

- You can split the multi-channel image into individual channels using F_VN_SplitImageChannels [749] and calculate a center of mass for each channel.
- You can convert the multi-channel image into a 1-channel image using F_VN_ConvertColorSpace [1083] and calculate a center of mass for it.

Center of mass (return value)

The calculated center of mass is returned via the reference aCenterOfMass.

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.13.16 F_VN_ImageCenterOfMassExp

Computes the center of mass of an image.

Syntax

Definition:

```plaintext
FUNCTION F_VN_ImageCenterOfMassExp : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    aCenterOfMass : Reference To TcVnPoint2_LREAL;
    ipMask : ITcVnImage;
    hrPrev : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ipSrcImage</code></td>
<td><code>ITcVnImage[383]</code></td>
<td>Source image (1 channel)</td>
</tr>
<tr>
<td><code>aCenterOfMass</code></td>
<td>Reference To <code>TcVnPoint2_LREAL[151]</code></td>
<td>Returns the center of mass of the image</td>
</tr>
<tr>
<td><code>ipMask</code></td>
<td><code>ITcVnImage[383]</code></td>
<td>Optional mask (1 channel of type USINT, same width and height as <code>ipSrcImage</code>)</td>
</tr>
<tr>
<td><code>hrPrev</code></td>
<td><code>HRESULT[135]</code></td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

`HRESULT[135]`

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.13.17 F_VN_ImageMedian

```c
FUNCTION F_VN_ImageMedian : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    aMedian    : Reference To TcVnVector4_LREAL;
    hrPrev     : HRESULT;
END_VAR
```

Computes the (approximated, channel-wise) median pixel value of an image.
Can use available TwinCAT Job Tasks for executing parallel code regions.

## Syntax

### Definition:

```c
FUNCTION F_VN_ImageMedian : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    aMedian    : Reference To TcVnVector4_LREAL;
    hrPrev     : HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ipSrcImage</code></td>
<td><code>ITcVnImage[383]</code></td>
<td>Source image (1 - 4 channels)</td>
</tr>
<tr>
<td><code>aMedian</code></td>
<td>Reference To <code>TcVnVector4_LREAL[153]</code></td>
<td>Returns the (approximated, channel-wise) median pixel value of <code>ipSrcImage</code></td>
</tr>
<tr>
<td><code>hrPrev</code></td>
<td><code>HRESULT[135]</code></td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.13.18  F_VN_ImageMedianExp

Computes the (approximated, channel-wise) median pixel value of an image. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

FUNCTION F_VN_ImageMedianExp : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  aMedian : Reference To TcVnVector4_LREAL;
  ipMask : ITcVnImage;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image (1 - 4 channels)</td>
</tr>
<tr>
<td>aMedian</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Returns the (approximated, channel-wise) median pixel value of ipSrcImage</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITcVnImage [383]</td>
<td>Mask of type USINT (1 channel)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base
**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.13.19 F_VN_ImageMoments

**F_VN_ImageMoments**

```plaintext
FUNCTION F_VN_ImageMoments: HRESULT
VAR_INPUT
    ipImage    : ITCvImage;
    stMoments  : Reference To TcVnMoments;
    bBinaryImage : BOOL;
    hrPrev     : HRESULT;
END_VAR
```

Computes the spatial moments, the central moments, and the central normalized moments of an image up to the third order.

#### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_ImageMoments : HRESULT
VAR_INPUT
    ipImage : ITCvImage;
    stMoments : Reference To TcVnMoments;
    bBinaryImage : BOOL;
    hrPrev : HRESULT;
END_VAR
```

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage</td>
<td>ITCvImage</td>
<td>Source image (1 channel, elements of type USINT or REAL)</td>
</tr>
<tr>
<td>stMoments</td>
<td>Reference To TcVnMoments</td>
<td>Returns a struct containing the moments</td>
</tr>
<tr>
<td>bBinaryImage</td>
<td>BOOL</td>
<td>If true, all non-zero pixels are treated as ones</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

#### Return value

**HRESULT [135]**

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.13.20 F_VN_MaxPixelValue

Finds the maximum pixel value in an image (1 - 4 channels supported).

Syntax

Definition:
FUNCTION F_VN_MaxPixelValue : HRESULT
VAR_INPUT
  ipImage : ITcVnImage;
  aMaxValue : Reference To TcVnVector4_LREAL;
  aPosition : Reference To TcVnPoint2_DINT;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>aMaxValue</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>Returns the maximum pixel value</td>
</tr>
<tr>
<td>aPosition</td>
<td>Reference To TcVnPoint2_DINT</td>
<td>Returns the first found position of aMaxValue</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.13.21 F_VN_MaxPixelValueExp
Finds the maximum pixel value in an image (1 - 4 channels supported). (expert function)

Syntax

**Definition:**

FUNCTION F_VN_MaxPixelValueExp : HRESULT
VAR_INPUT
  ipImage : ITcVnImage;
  aMaxValue : Reference To TcVnVector4_LREAL;
  aPosition : Reference To TcVnPoint2_DINT;
  ipMask : ITcVnImage;
  eVectorCompareMethod : ETcVnVectorCompareMethod;
  hrPrev : HRESULT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage</td>
<td>ITcVnImage</td>
<td>Source image</td>
</tr>
<tr>
<td>aMaxValue</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>Returns the maximum pixel value</td>
</tr>
<tr>
<td>aPosition</td>
<td>Reference To TcVnPoint2_DINT</td>
<td>Returns the first found position of aMaxValue (not supported for multi-channel images with ELEMENTWISE)</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITcVnImage</td>
<td>Optional mask to specify which pixel positions are considered (USINT, set parameter to 0 if not required)</td>
</tr>
<tr>
<td>eVectorCompareMethod</td>
<td>ETcVnVectorCompareMethod</td>
<td>Select a vector compare method for multi-channel images</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.13.22 F_VN_MinPixelValue

Finds the minimum pixel value in an image (1 - 4 channels supported).
Syntax

Definition:

FUNCTION F_VN_MinPixelValue : HRESULT
VAR_INPUT
    ipImage     : ITcVnImage;
    aMinValue   : Reference To TcVnVector4_LREAL;
    aPosition   : Reference To TcVnPoint2_DINT;
    hrPrev      : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage</td>
<td>ITcVnImage</td>
<td>Source image</td>
</tr>
<tr>
<td>aMinValue</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>Returns the minimum pixel value</td>
</tr>
<tr>
<td>aPosition</td>
<td>Reference To TcVnPoint2_DINT</td>
<td>Returns the first found position of aMinValue</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.13.23 F_VN_MinPixelValueExp

Finds the minimum pixel value in an image (1 - 4 channels supported). (expert function)

Syntax

Definition:

FUNCTION F_VN_MinPixelValueExp : HRESULT
VAR_INPUT
    ipImage     : ITcVnImage;
    aMinValue   : Reference To TcVnVector4_LREAL;
    aPosition   : Reference To TcVnPoint2_DINT;
    ipMask      : ITcVnImage;
    eVectorCompareMethod : ETCvVectorCompareMethod;
    hrPrev      : HRESULT;
END_VAR
API reference

<table>
<thead>
<tr>
<th>eVectorCompareMethod</th>
<th>ETcVnVectorCompareMethod</th>
<th>hrPrev</th>
<th>HRESULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>END_VAR</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>aMinValue</td>
<td>Reference To TcVnVector4_LREAL [153]</td>
<td>Returns the minimum pixel value</td>
</tr>
<tr>
<td>aPosition</td>
<td>Reference To TcVnPoint2_DINT [151]</td>
<td>Returns the first found position of aMinValue (not supported for multi-channel images with ELEMENTWISE)</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITcVnImage [383]</td>
<td>Optional mask to specify which pixel positions are considered (USINT, set parameter to 0 if not required)</td>
</tr>
<tr>
<td>eVectorCompareMethod</td>
<td>ETcVnVectorCompareMethod [202]</td>
<td>Select a vector compare method for multi-channel images</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.14 Image Color and Contrast Processing

This group contains functions for color and contrast processing.

**Functions**

Frequency distribution, intensity and contrast Adjustment of an image

- F_VN_Clahe(Exp) [1081]
- F_VN_Histogram(Exp) [1087]
- F_VN_HistogramEqualization(Exp) [1088]
- F_VN_NormalizeImage(Exp) [1094]
- F_VN_NormalizeImageForDisplay [1096]

Look-Up Table and Color Maps

- F_VN_ApplyColorMap [1079]
- F_VN_ApplyLut [1080]
- F_VN_GenerateColorMap [1084]
• F_VN_GenerateCustomColorMap [► 1086]

Color Matching
• F_VN_ReferenceColorSimilarity(Exp) [► 1097]
• F_VN_TrainImageColor(Exp) [► 1102]

Miscellaneous
• F_VN_ConvertColorSpace [► 1083]
• F_VN_InvertImageColor(Exp) [► 1092]

Samples
Image Color and Contrast Processing [► 1410]

6.4.14.1 F_VN_ApplyColorMap

Apply a color map to a gray-level image.

Syntax
Definition:
FUNCTION F_VN_ApplyColorMap : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  ipColorMap : ITcVnContainer;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [► 383]</td>
<td>Source image (TCVN_ET_USINT or TCVN_ET_UINT, 1 channel)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Returns the colored image (TCVN_ET_USINT, 3 channel. An appropriate image will be created if required.)</td>
</tr>
<tr>
<td>ipColorMap</td>
<td>ITcVnContainer [► 345]</td>
<td>Color map to be applied to ipSrcImage (ContainerType_Vector_TcVnVector3_REAL with 256 or 65536 elements, dependent on ipSrcImage type). Can be either custom or created with F_VN_GenerateColorMap.</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [► 135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [► 135]

Further information

The function F_VN_ApplyColorMap applies a color table to the input image.
Parameter

Input image

The input image `ipSrcImage` must be a 1-channel grayscale image and have an element type of `USINT` (8-bit) or `UINT` (16-bit).

Result image

The result image `ipDestImage` returns the 3-channel RGB color image with the element type `USINT` (8-bit) that was transformed with the color table.

Color table

The color table `ipColorMap` defines how the input image is transformed. The color table is a container that can be created either manually or with one of the following two functions:

- `F_VN_GenerateColorMap` [1084]
- `F_VN_GenerateCustomColorMap` [1086]

The container must contain 256 elements (for 8-bit) or 65,536 elements (for 16-bit), depending on the element type of the input image, and must be of the type `ContainerType_Vector_TcVnVector3_REAL`.

Application

The application of a typical heat map to an 8-bit image looks like this, for example:

```plaintext
hr := F_VN_GenerateColorMap(ipColorMap, TCVN_CMM_HOT, TCVN_CMS_256, hr);
hr := F_VN_ApplyColorMap(
    ipSrcImage := ipImageIn,
    ipDestImage := ipImageRes,
    ipColorMap := ipColorMap,
    hrPrev := hr
);
```

Samples

- Display with color tables [1413]

Related functions

- `F_VN_GenerateColorMap` [1084]
- `F_VN_GenerateCustomColorMap` [1086]
- `F_VN_ApplyColorMap` [1079]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.14.2 F_VN_ApplyLut

```
F_VN_ApplyLut
___ ipSrcImage  ITcVnImage  HRESULT  F_VN_ApplyLut
___ ipDestImage Reference To ITcVnImage
___ ipLut       ITcVnContainer
___ hrPrev     HRESULT
```
Apply a lookup table to an image to manipulate its colors.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_ApplyLut : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    ipLut : ITcVnContainer;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image (1-4 channels of 8 or 16 bit types SINT, USINT, INT, UINT)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Returns the result image (same size, type and channels as ipSrcImage. An appropriate image will be created if required.)</td>
</tr>
<tr>
<td>ipLut</td>
<td>ITcVnContainer [345]</td>
<td>Lookup table with elements matching ipSrcImage type and either 256 (8 bit images) or 65536 (16 bit images) elements. E.g. ContainerType_Vector_USINT for 1-4 channel USINT image (in this case, the same lookup values are used for each channel) or ContainerType_Vector_TcVnVector3_USINT for a 3 channel USINT image (in this case, each channel uses an individual lookup value).</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

```
HRESULT [135]
```

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.14.3 F_VN_Clahe**

```
F_VN_Clahe
    ipSrcImage     ITcVnImage
    ipDestImage    Reference To ITcVnImage
    hrPrev         HRESULT
```

Apply Contrast Limited Adaptive Histogram Equalization. This applies local histogram equalization to individual tiles.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Syntax

Definition:

FUNCTION F_VN_Clahe : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image (TCVN_ET_USINT or TCVN_ET_UINT, 1 channel)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Returns the resulting image (An appropriate image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.14.4  F_VN_ClaheExp

F_VN_ClaheExp

Apply Contrast Limited Adaptive Histogram Equalization. This applies local histogram equalization to individual tiles. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

FUNCTION F_VN_ClaheExp : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    fClipLimit  : LREAL;
    nTilesX     : UDINT;
    nTilesY     : UDINT;
    hrPrev      : HRESULT;
END_VAR
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [138]</td>
<td>Source image (TCVN_ET_USINT or TCVN_ET_UINT, 1 channel)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [138]</td>
<td>Returns the resulting image (An appropriate image will be created if required.)</td>
</tr>
<tr>
<td>fClipLimit</td>
<td>LREAL</td>
<td>Threshold for contrast limit (set to &lt;= 0 to disable clipping)</td>
</tr>
<tr>
<td>nTilesX</td>
<td>UDINT</td>
<td>Number of tiles in x direction</td>
</tr>
<tr>
<td>nTilesY</td>
<td>UDINT</td>
<td>Number of tiles in y direction</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.14.5  F_VN_ConvertColorSpace**

Convert image from one color space to another.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

**Syntax**

**Definition:**

FUNCTION F_VN_ConvertColorSpace : HRESULT
VAR_INPUT
   ipSrcImage : ITcVnImage;
   ipDestImage : Reference To ITcVnImage;
   eTransform : ETcVnColorSpaceTransform;
   hrPrev : HRESULT;
END_VAR
API reference

## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>eTransform</td>
<td>ETcVnColorSpaceTransform [163]</td>
<td>Transforms to be applied</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.14.6 F_VN_GenerateColorMap

Generates a pre-defined color map.

### Syntax

**Definition:**

```c
FUNCTION F_VN_GenerateColorMap : HRESULT
VAR_INPUT
    ipColorMap  : Reference To ITcVnContainer;
    eColorMap   : ETcVnColorMap;
    eColorMapSize : ETcVnColorMapSize;
    hrPrev      : HRESULT;
END_VAR
```
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipColorMap</td>
<td>Reference To ITcVnContainer</td>
<td>Returns the color map (ContainerType_Vector_TcVnVector3_REAL with 256 or 65536 elements)</td>
</tr>
<tr>
<td>eColorMap</td>
<td>ETcVnColorMap [160]</td>
<td>Selects a color map (similar to GNU Octave/MATLAB types)</td>
</tr>
<tr>
<td>eColorMapSize</td>
<td>ETcVnColorMapSize [162]</td>
<td>Defines how many elements the generated color map should have</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Further information

The function F_VN_GenerateColorMap creates a color table that can be applied to an image with the function F_VN_ApplyColorMap [1079].

Parameter

Color Map

The parameter ipColorMap returns the color table created.

Color gradient

The color gradient eColorMap of the color table is defined via the enum ETcVnColorMap [160]. If you wish to define a color gradient manually, use the function F_VN_GenerateCustomColorMap [1086].

Size of the color table

The size eColorMapSize is specified via the enum ETcVnColorMapSize [162] and defines whether the color table is created for 8-bit images (TCVN_CMS_256) or for 16-bit images (TCVN_CMS_65536).

Application

```
VAR
  ipColorMap : ITcVnColorMap;
END_VAR

hr := F_VN_GenerateColorMap(
  ipColorMap := ipColorMap,
  eColorMap := TCVN_CM_HOT,
  eColorMapSize := TCVN_CMS_256,
  hrPrev := hr);
```

Samples

- Display with color tables [1413]

Related functions

- F_VN_GenerateColorMap [1084]
- F_VN_GenerateCustomColorMap [1086]
- F_VN_ApplyColorMap [1079]
**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.14.7  F_VN_GenerateCustomColorMap

Generate a custom color map by interpolating between user defined colors (equally distributed, linear for each channel).

**Syntax**

**Definition:**

```c
FUNCTION F_VN_GenerateCustomColorMap : HRESULT
VAR_INPUT
 ipColorMap : Reference To ITcVnContainer;
 ipInitialColors : ITcVnContainer;
 eColorMapSize : ETcVnColorMapSize;
 hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipColorMap</td>
<td>Reference To ITcVnContainer</td>
<td>Returns the color map (ContainerType_Vector_TcVnVector3_REAL with 256 or 65536 elements)</td>
</tr>
<tr>
<td>ipInitialColors</td>
<td>ITcVnContainer</td>
<td>User defined colors (ContainerType_Vector_TcVnVector3_REAL, at least 2 elements)</td>
</tr>
<tr>
<td>eColorMapSize</td>
<td>ETcVnColorMapSize</td>
<td>Defines how many elements the generated color map should have</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Samples**

- Display with color tables [1413]

**Related functions**

- F_VN_GenerateColorMap [1084]
- F_VN_GenerateCustomColorMap [1086]
• F_VN_ApplyColorMap [1079]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.14.8 F_VN_Histogram

Calculate the (multi-channel) histogram of an image. Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:
FUNCTION F_VN_Histogram : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  ipDestHistogram : Reference To ITcVnContainer;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestHistogram</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container with a multi-channel histogram, where every channel is represented as a vector of UDINT (ContainerType_Vector_Vector_UDINT. Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Further information

The function F_VN_Histogram calculates the histogram of the input image channel by channel.

Parameter

Input image

The input image ipSrcImage can have any format.
Histogram

The calculated multichannel histogram is returned as a container ipDestHistogram of the type ContainerType Vector Vector UDINT. Each sub-element corresponds to the histogram of one channel. Each container of the histogram is represented as a UDINT, which specifies the number of pixels belonging to the container.

Expert parameters

The expert version F_VN_HistogramExp [1090] contains additional parameters.

Application

The determination of the most frequently occurring pixel value in a grayscale image looks like this, for example:

```plaintext
VAR
    ipDestHistograms : ITcVnContainer;
    ipSingleHistogram : ITcVnContainer;
    nIndex            : ULINT;
    ipIterator        : ITcVnBidirectionalIterator;
END_VAR

hr := F_VN_Histogram(
    ipSrcImage      := ipImageIn,
    ipDestHistogram := ipHistograms,
    hrPrev          := hr);
hr := F_VN_GetAt_ITcVnContainer(ipHistograms, ipSingleHistogram, 0, hr);
hr := F_VN_MaxElement(ipSingleHistogram, ipIterator, nIndex, hr);
// nIndex now contains the most frequent pixel value in ipImageIn.
```

Samples

- Handling histograms

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.14.9 F_VN_HistogramEqualization

Equalize the histogram of a grayscale or rgb image, which normalizes the brightness and improves the contrast.

Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```plaintext
FUNCTION F_VN_HistogramEqualization : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev : HRESULT;
```

```plaintext
F_VN_HistogramEqualization
ip SrcImage  ITcVnImage  HRESULT F_VN_HistogramEqualization
ip DestImage Reference To ITcVnImage
hrPrev  HRESULT
```
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.14.10 F_VN_HistogramEqualizationExp

Equalize the histogram of a grayscale or color image, which normalizes the brightness and improves the contrast. (expert function) The channel index that should be equalized has to be specified (-1 expects a RGB image, converts it to YCbCr, equalizes the Y channel and converts the image back to RGB).

Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_HistogramEqualizationExp : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    nChannelIdx : INT;
    hrPrev : HRESULT;
END_VAR
```
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>nChannelIdx</td>
<td>INT</td>
<td>Index of the image channel that should be equalized (-1 expects a RGB image, converts it to YCbCr, equalizes the Y channel and converts the image back to RGB)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.14.11 F_VN_HistogramExp

Calculate the (multi-channel) histogram of an image. (expert function)

Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_HistogramExp : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipDestHistogram : Reference To ITcVnContainer;
    nBins         : Reference To UDINT;
    fLowerBound   : Reference To LREAL;
    fUpperBound   : Reference To LREAL;
    ipMask        : ITcVnImage;
    hrPrev        : HRESULT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestHistogram</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container with a multi-channel histogram, where every channel is represented as a vector of UDINT (ContainerType_Vector_Vector_UDINT. Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>nBins</td>
<td>Reference To UDINT</td>
<td>Desired number of bins or 0 to keep the default for the corresponding image format (in) and default number of bins (out)</td>
</tr>
<tr>
<td>fLowerBound</td>
<td>Reference To LREAL</td>
<td>Lower (inclusive) boundary of the 0-th histogram bin (in), or receive the default if fLowerBound AND fUpperBound are set to 0 (out)</td>
</tr>
<tr>
<td>fUpperBound</td>
<td>Reference To LREAL</td>
<td>Upper (exclusive) boundary of the last histogram bin nBins-1 (in), or receive the default if fLowerBound AND fUpperBound are set to 0 (out)</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITcVnImage [383]</td>
<td>Source image mask (TCVN_ET_USINT, 1 channel. Mask is optional, set to 0 if not required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Further information**

The function F_VN_HistogramExp is the expert version of F_VN_Histogram [1087]. It contains additional parameters.

**Parameter**

**Input image**

The input image ipSrcImage can have any format.

**Histogram**

The calculated multichannel histogram is returned as a container ipDestHistogram of the type ContainerType_Vector_Vector_UDINT. Each sub-element corresponds to the histogram of one channel. Each container of the histogram is represented as a UDINT, which specifies the number of pixels belonging to the container.

**Container**

The number nBins defines how many containers the observed pixel spectrum is to be divided into.

**Limits**

The lower limit fLowerBound and the upper limit fUpperBound define which pixel values are taken into account when creating the histogram.
Mask

The mask \texttt{ipMask} defines which pixels of the input image \texttt{ipSrcImage} are taken into account in the calculation of the histogram. It is described by a 1-channel image with the element type \texttt{USINT}, which has the same size as the input image \texttt{ipSrcImage}. All pixels in the mask image with a value \texttt{<> 0} are taken into account in the calculation of the histogram.

Application

The calculation of a histogram with 10 containers for the value range 100-200 with a circular mask looks like this, for example:

```c
VAR
    ipImageMask : ITCvImage;
END_VAR

// Create mask image with a circle at image center.
hr := F_VN_CopyImage(ipImageIn, ipImageMask, hr);
hr := F_VN_SetPixels(ipImageMask, aBlack, hr);
hr := F_VN_DrawCircle(400, 300, 300, ipImageMask, aWhite, -1, hr);

hr := F_VN_HistogramExp(
    ipSrcImage := ipImageIn,
    ipDestHistogram := ipHistograms,
    nBins := 10,
    fLowerBound := 100,
    fUpperBound := 201,
    ipMask := ipImageMask,
    hrPrev := hr
);
```

Samples

- Handling histograms

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.14.12 \texttt{F_VN_InvertImageColor}

Invert the color of an image. For signed integer or negative floating point values, only the signs will be switched. If the image contains only positive floating point values, each pixel value is subtracted from the maximum available pixel value (or 1.0, whatever is higher).

Syntax

```
FUNCTION F_VN_InvertImageColor : HRESULT
VAR_INPUT
    ipSrcImage : ITCvImage;
    ipDestImage : Reference To ITCvImage;
    hrPrev     : HRESULT;
END_VAR
```

Invert the color of an image. For signed integer or negative floating point values, only the signs will be switched. If the image contains only positive floating point values, each pixel value is subtracted from the maximum available pixel value (or 1.0, whatever is higher).
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.14.13  F_VN_InvertImageColorExp

Invert the color of an image. (expert function) For signed integer or negative floating point values, only the signs will be switched. If the image contains only positive floating point values, each pixel value is subtracted from the maximum available pixel value (or 1.0, whatever is higher).

Syntax

Definition:

```plaintext
FUNCTION F_VN_InvertImageColorExp : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    fMaxValue : LREAL;
    hrPrev : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate image will be created if required.)</td>
</tr>
<tr>
<td>fMaxValue</td>
<td>LREAL</td>
<td>Maximum pixel value (e.g. if a 16 bit image contains 12 bit values). -1 means the default values are used.</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
### Return value

HRESULT [\ref 135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

#### 6.4.14.14 F_VN_NormalizeImage

```plaintext
F_VN_NormalizeImage
ipSrcImage ITCvNImage
ipDestImage Reference To ITCvNImage
hrPrev HRESULT
```

Normalize an image regarding its value range (e.g. stretch pixel values [50..150] to full range [0..255]).

### Syntax

#### Definition:

```plaintext
FUNCTION F_VN_NormalizeImage : HRESULT
VAR_INPUT
  ipSrcImage : ITCvNImage;
  ipDestImage : Reference To ITCvNImage;
  hrPrev : HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvNImage[\ref 383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITCvNImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [\ref 135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [\ref 135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
F_VN_NormalizeImageExp

Normalize an image regarding its value range (e.g. stretch pixel values [50..150] to full range [0..255]) or scale the values regarding a specific normalization (e.g. L2-norm). (expert function)

Syntax

Definition:

```c
FUNCTION F_VN_NormalizeImageExp : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    fAlpha : LREAL;
    fBeta : LREAL;
    eNormType : ETcVnNormalizationType;
    eDestType : ETcVnElementType;
    ipMask : ITcVnImage;
    hrPrev : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage[383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>fAlpha</td>
<td>LREAL</td>
<td>Lower range boundary (in case of normalizing the value range) or value to normalize to (</td>
</tr>
<tr>
<td>fBeta</td>
<td>LREAL</td>
<td>Upper range boundary (in case of normalizing the value range)</td>
</tr>
<tr>
<td>eNormType</td>
<td>ETcVnNormalizationType [195]</td>
<td>Normalization type (only INF, L1, L2 or MINMAX)</td>
</tr>
<tr>
<td>eDestType</td>
<td>ETcVnElementType[186]</td>
<td>Destination image depth (usually SAME_AS_SOURCE)</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITcVnImage[383]</td>
<td>Mask to restrict the normalization to specific pixel positions (set 0 to normalize the whole image)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT[135]

Required License

TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.14.16 F_VN_NormalizeImageForDisplay

**F_VN_NormalizeImageForDisplay**

```
ipSrcImage : ITCvnImage;
ipDestImage : Reference To ITCvnImage;
eSignedNormalization : ETcVnSignedNormalization;
hrPrev : HRESULT;
```

Normalize an image for display, i.e. scale it to the full value range of the underlying data type (-1 to 1 for floating point).
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

**Syntax**

**Definition:**

```
FUNCTION F_VN_NormalizeImageForDisplay : HRESULT
VAR_INPUT
    ipSrcImage : ITCvnImage;
    ipDestImage : Reference To ITCvnImage;
    eSignedNormalization : ETcVnSignedNormalization;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITCvImage [383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>eSignedNormalization</td>
<td>ETcVnSignedNormalization [199]</td>
<td>Option for normalizing signed values</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

**HRESULT [135]**

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.14.17  F_VN_ReferenceColorSimilarity_ITcVnColorModel

Computes the similarity to a reference color model for each pixel in the source image.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

FUNCTION F_VN_ReferenceColorSimilarity_ITcVnColorModel : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    ipColorModel : ITcVnColorModel;
    hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>RGB source image (USINT, 3 channels)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Returns the similarity to ipColorModel for each pixel in ipSrcImage (USINT, 1 channel. An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>ipColorModel</td>
<td>ITcVnColorModel</td>
<td>Color model</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Related functions

• F_VN_Threshold [1166] for binary segmentation on the basis of a threshold value
• F_VN_AdaptiveThreshold [1162] for binary, adaptive segmentation
• F_VN_CheckColorRange [1165] for binary segmentation on the basis of a color spectrum
• F_VN_ReferenceColorSimilarity [1097] for binary segmentation on the basis of a color model

Related functions

• F_VN_TrainImageColor [1102] for the training of a color model
• F_VN_ReferenceColorSimilarity [1097] for segmentation by means of a color model

Required License

TC3 Vision Base
### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.14.18 F_VN_ReferenceColorSimilarity_ITcVnMlModel

Computes the similarity to a reference color model for each pixel in the source image. Can use available TwinCAT Job Tasks for executing parallel code regions. Can return partial results when canceled by Watchdog.

#### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_ReferenceColorSimilarity_ITcVnMlModel : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    ipColorModel : ITcVnMlModel;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>RGB source image (USINT, 3 channels)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Returns the similarity to ipColorModel for each pixel in ipSrcImage (USINT, 1 channel. An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>ipColorModel</td>
<td>ITcVnMlModel [392]</td>
<td>Color model</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

**RESULT [135]**

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.14.19 F_VN_ReferenceColorSimilarity_TcVnVector3_LREAL

Computes the similarity to a reference color for each pixel in the source image. Can use available TwinCAT Job Tasks for executing parallel code regions. Can return partial results when canceled by Watchdog.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_ReferenceColorSimilarity_TcVnVector3_LREAL : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    aRefColor : Reference To TcVnVector3_LREAL;
    hrPrev : HRESULT;
END_VAR

HRESULT F_VN_ReferenceColorSimilarity_TcVnVector3_LREAL
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>RGB source image (USINT, 3 channels)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Returns the similarity to aRefColor for each pixel in ipSrcImage (USINT, 1 channel. An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>aRefColor</td>
<td>Reference To TcVnVector3_LREAL</td>
<td>Reference color (RGB, [0..255])</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.14.20 F_VN_ReferenceColorSimilarityExp_ITcVnColorModel
Computes the similarity to a reference color model for each pixel in the source image. (expert function) Can use available TwinCAT Job Tasks for executing parallel code regions. Can return partial results when canceled by Watchdog.

### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_ReferenceColorSimilarityExp_ITcVnColorModel : HRESULT
VAR_INPUT
   ipSrcImage : ITcVnImage;
   ipDestImage : Reference To ITcVnImage;
   ipColorModel : ITcVnColorModel;
   fVariance : REAL;
   fLuminanceWeight : REAL;
   hrPrev : HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>RGB source image (USINT, 3 channels)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Returns the similarity to ipColorModel for each pixel in ipSrcImage (USINT, 1 channel. An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>ipColorModel</td>
<td>ITcVnColorModel [391]</td>
<td>Color model</td>
</tr>
<tr>
<td>fVariance</td>
<td>REAL</td>
<td>Allowed color variance (0.1 - 0.3 might be a good start to try)</td>
</tr>
<tr>
<td>fLuminanceWeight</td>
<td>REAL</td>
<td>Weight the impact of the luminance ([0..1], e.g. set to 0 to be more resistant to unequal illumination, but might be required to differentiate between some colors. Ignored if ipColorModel is of type RGB.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

**HRESULT [135]**

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.14.21 **F_VN_ReferenceColorSimilarityExp_ITcVnMIModel**
Computes the similarity to a reference color model for each pixel in the source image. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:
FUNCTION F_VN_ReferenceColorSimilarityExp_ITcVnMlModel : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  ipColorModel : ITcVnMlModel;
  fVariance : REAL;
  fLuminanceWeight : REAL;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>RGB source image (USINT, 3 channels)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Returns the similarity to ipColorModel for each pixel in ipSrcImage (USINT, 1 channel. An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>ipColorModel</td>
<td>ITcVnMlModel [392]</td>
<td>Color model</td>
</tr>
<tr>
<td>fVariance</td>
<td>REAL</td>
<td>Allowed color variance (0.1 - 0.3 might be a good start to try)</td>
</tr>
<tr>
<td>fLuminanceWeight</td>
<td>REAL</td>
<td>Weight the impact of the luminance ([0..1], e.g. set to 0 to be more resistant to unequal illumination, but might be required to differentiate between some colors. Ignored if ipColorModel is of type RGB.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT \[135\]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.14.22 F_VN_ReferenceColorSimilarityExp_TcVnVector3_ LREAL

Computes the similarity to a reference color for each pixel in the source image. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.
API reference

Syntax

Definition:

FUNCTION F_VN_ReferenceColorSimilarityExp_TcVnVector3_LREAL : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    aRefColor : Reference To TcVnVector3_LREAL;
    fVariance : REAL;
    fLuminanceWeight : REAL;
    hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [135]</td>
<td>RGB source image (USINT, 3 channels)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [135]</td>
<td>Returns the similarity to aRefColor for each pixel in ipSrcImage (USINT, 1 channel. An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>aRefColor</td>
<td>Reference To TcVnVector3_LREAL [153]</td>
<td>Reference color (RGB, [0..255])</td>
</tr>
<tr>
<td>fVariance</td>
<td>REAL</td>
<td>Allowed color variance (0.1 - 0.3 might be a good start to try)</td>
</tr>
<tr>
<td>fLuminanceWeight</td>
<td>REAL</td>
<td>Weight the impact of the luminance ([0..1], e.g. set to 0 to be more resistant to unequal illumination, but might be required to differentiate between some colors)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Further information

Samples

- [Color similarity with RGB reference color](#)

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.14.23 F_VN_TrainImageColor

[F_VN_TrainImageColor](#)
Create a new color model, describing the image color. Can use available TwinCAT Job Tasks for executing parallel code regions. Can return partial results when canceled by Watchdog.

**Syntax**

**Definition:**

```
FUNCTION F_VN_TrainImageColor : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipColorModel : Reference To ITcVnColorModel;
    nDifferentColors : UDINT;
    hrPrev : HRESULT;
END_VAR
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image (3 channels (RGB) of type USINT)</td>
</tr>
<tr>
<td>ipColorModel</td>
<td>Reference To ITcVnColorModel</td>
<td>Returns the color model</td>
</tr>
<tr>
<td>nDifferentColors</td>
<td>UDINT</td>
<td>Number of different colors to distinguish</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Further information**

The function **F_VN_TrainImageColor** trains a color model on the basis of a reference image.

**Parameter**

**Reference image**

The reference image **ipSrcImage** must have 3 channels in RGB format and the element type **USINT** (8-bit).

**Color model**

The parameter **ipColorModel** returns the trained color model. This is an interface pointer of the type **ITcVnColorModel** [391].

**Number of colors to be distinguished**

The number **nDifferentColors** defines the number of colors to be trained. Note that the image background, for example, can also have its own color.

**Expert parameters**

Further parameters can be found in the expert version **F_VN_TrainImageColorExp** [1105].

**Application**

The training of a color model for a reference image with 4 differently colored objects and a dark background looks like this, for example:
VAR
    ipColorModel : ITCvNColorModel;
END_VAR

hr := F_VN_TrainImageColor{
    ipSrcImage := ipImageRef,
    ipColorModel := ipColorModel,
    nDifferentColors := 5, // 4 colors + 1 background = 5
    hrPrev := hr
};

Related functions

• F_VN_TrainImageColor [1102] for the training of a color model
• F_VN_ReferenceColorSimilarity [1097] for segmentation by means of a color model

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.14.24 F_VN_TrainImageColor_ITcVnMlModel

Create a new color model, describing the image color.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:
FUNCTION F_VN_TrainImageColor_ITcVnMlModel : HRESULT
VAR_INPUT
    ipSrcImage : ITCvNImage;
    ipColorModel : Reference To ITCvNModel;
    nDifferentColors : UDINT;
    hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvNImage [383]</td>
<td>Source image (3 channels (RGB) of type USINT)</td>
</tr>
<tr>
<td>ipColorModel</td>
<td>Reference To ITCvNModel [392]</td>
<td>Returns the color model</td>
</tr>
<tr>
<td>nDifferentColors</td>
<td>UDINT</td>
<td>Number of different colors to distinguish</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g.</td>
<td>Tc3_Vision</td>
</tr>
<tr>
<td></td>
<td>Intel 4-core Atom CPU</td>
<td></td>
</tr>
</tbody>
</table>

6.4.14.25  F_VN_TrainImageColorExp

Create a new color model, describing the image color. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

FUNCTION F_VN_TrainImageColorExp : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipColorModel : Reference To ITcVnColorModel;
    nDifferentColors : UDINT;
    eMethod : ETcVnColorTrainingMethod;
    ipMask : ITcVnImage;
    nSkipPixels : UDINT;
    hrPrev : HRESULT;
END_VAR
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage ![383]</td>
<td>Source image (3 channels (RGB) of type USINT)</td>
</tr>
<tr>
<td>ipColorModel</td>
<td>Reference To ITcVnColorModel ![391]</td>
<td>Returns the color model</td>
</tr>
<tr>
<td>nDifferentColors</td>
<td>UDINT</td>
<td>Number of different colors to distinguish</td>
</tr>
<tr>
<td>eMethod</td>
<td>ETcVnColorTrainingMethod ![174]</td>
<td>Color training method</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITcVnImage ![383]</td>
<td>Optional image mask (1 channel of type USINT, set to 0 if not required)</td>
</tr>
<tr>
<td>nSkipPixels</td>
<td>UDINT</td>
<td>Number of pixels to skip between each evaluated color sample (to achieve a better performance). 0 takes every pixel into account and tends to be more accurate.</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT ![135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT ![135]

### Further information

The function F_VN_TrainImageColorExp is the expert version of F_VN_TrainImageColor ![1102]. It contains further parameters.

### Parameter

**Reference image**

The reference image ipSrcImage must have 3 channels in RGB format and the element type USINT (8-bit).

**Color model**

The parameter ipColorModel returns the trained color model. This is an interface pointer of the type ITcVnColorModel ![391].

**Number of colors to be distinguished**

The number nDifferentColors defines the number of colors to be trained. Note that the image background, for example, can also have its own color.

**Training methods**

**Mask**

**Pixel gap**

**Application**

```vb
VAR
    ipColorModel : ITcVnColorModel;
END_VAR
/hr := F_VN_TrainImageColor;
```
ipSrcImage := ipImageRef,
ipColorModel := ipColorModel,
nDifferentColors := 5,  // 4 colors + 1 background = 5
eMethod := TCVN_CTM_LAB,
ipMask := 0,
nSkipPixels := 3,
hrPrev := hr
);

Related functions

• F_VN_TrainImageColor [ ] for the training of a color model

• F_VN_ReferenceColorSimilarity [ ] for segmentation by means of a color model

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.1426 F_VN_TrainImageColorExp_ITcVnMlModel

Create a new color model, describing the image color. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

FUNCTION F_VN_TrainImageColorExp_ITcVnMlModel : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipColorModel : Reference To ITcVnMlModel;
    nDifferentColors : UDINT;
    eMethod : ETCVNColorTrainingMethod;
    ipMask : ITcVnImage;
    nSkipPixels : UDINT;
    hrPrev : HRESULT;
END_VAR
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image (3 channels (RGB) of type USINT)</td>
</tr>
<tr>
<td>ipColorModel</td>
<td>Reference To ITcVnColorModel [392]</td>
<td>Returns the color model</td>
</tr>
<tr>
<td>nDifferentColors</td>
<td>UDINT</td>
<td>Number of different colors to distinguish</td>
</tr>
<tr>
<td>eMethod</td>
<td>ETCvNColorTrainingMethod [174]</td>
<td>Color training method</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITcVnImage [383]</td>
<td>Optional image mask (1 channel of type USINT, set to 0 if not required)</td>
</tr>
<tr>
<td>nSkipPixels</td>
<td>UDINT</td>
<td>Number of pixels to skip between each evaluated color sample (to achieve a better performance). 0 takes every pixel into account and tends to be more accurate.</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.14.27 F_VN_TrainImageColorExp2

Create a new color model, describing the image color. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_TrainImageColorExp2 : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
```
ipColorModel : Reference To ITcVnColorModel;
nDifferentColors : UDINT;
eMethod : ETcVnColorTrainingMethod;
ipMask : ITcVnImage;
nSkipPixels : UDINT;
eClusteringAlgorithm : ETcVnClusteringAlgorithm;
nMaxClusterRadius : DINT;
bSingleSplitSteps : BOOL;
hrPrev : HRESULT;
END_VAR

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image (3 channels (RGB) of type USINT)</td>
</tr>
<tr>
<td>ipColorModel</td>
<td>Reference To ITcVnColorModel [391]</td>
<td>Returns the color model</td>
</tr>
<tr>
<td>nDifferentColors</td>
<td>UDINT</td>
<td>Maximum number of different colors to distinguish (if LBG is used as a clustering algorithm, the result might have less different colors, depending on nMaxClusterRadius)</td>
</tr>
<tr>
<td>eMethod</td>
<td>ETcVnColorTrainingMethod [174]</td>
<td>Color training method</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITcVnImage [383]</td>
<td>Optional image mask (1 channel of type USINT, set to 0 if not required)</td>
</tr>
<tr>
<td>nSkipPixels</td>
<td>UDINT</td>
<td>Number of pixels to skip between each evaluated color sample (to achieve a better performance). 0 takes every pixel into account and tends to be more accurate.</td>
</tr>
<tr>
<td>eClusteringAlgorithm</td>
<td>ETcVnClusteringAlgorithm [159]</td>
<td>Clustering algorithm</td>
</tr>
<tr>
<td>nMaxClusterRadius</td>
<td>DINT</td>
<td>Only used for the LBG clustering algorithm. Maximum allowed radius (&gt; 0) of a single cluster, i.e. clusters with a higher radius will be split into smaller ones, until a global number of nDifferentColors is reached.</td>
</tr>
<tr>
<td>bSingleSplitSteps</td>
<td>BOOL</td>
<td>Only used for the LBG clustering algorithm. If true, the global optimization is always run after a single cluster has been split. If false, several clusters are split within the same step before applying the global optimization. Applying the global optimization less often is faster, but can lead to less optimal results, especially having 2 nearby clusters that could be represented by 1.</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
Create a new color model, describing the image color. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

**Syntax**

**Definition:**

```
FUNCTION F_VN_TrainImageColorExp2_ITcVnMlModel : HRESULT
VAR_INPUT
    ipSrcImage : ITCvImage;
    ipColorModel : Reference To ITCvMlModel;
    nDifferentColors : UDINT;
    eMethod : ETcVnColorTrainingMethod;
    ipMask : ITCvImage;
    nSkipPixels : UDINT;
    eClusteringAlgorithm : ETcVnClusteringAlgorithm;
    nMaxClusterRadius : DINT;
    bSingleSplitSteps : BOOL;
    hrPrev : HRESULT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [138]</td>
<td>Source image (3 channels (RGB) of type USINT)</td>
</tr>
<tr>
<td>ipColorModel</td>
<td>Reference To ITcVnMlModel [139]</td>
<td>Returns the color model</td>
</tr>
<tr>
<td>nDifferentColor</td>
<td>UDINT</td>
<td>Maximum number of different colors to distinguish (if LBG is used as a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>clustering algorithm, the result might have less different colors, depending</td>
</tr>
<tr>
<td></td>
<td></td>
<td>on nMaxClusterRadius)</td>
</tr>
<tr>
<td>eMethod</td>
<td>ETcVnColorTrainingMethod [140]</td>
<td>Color training method</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITcVnImage [138]</td>
<td>Optional image mask (1 channel of type USINT, set to 0 if not required)</td>
</tr>
<tr>
<td>nSkipPixels</td>
<td>UDINT</td>
<td>Number of pixels to skip between each evaluated color sample (to achieve</td>
</tr>
<tr>
<td></td>
<td></td>
<td>a better performance). 0 takes every pixel into account and tends to be</td>
</tr>
<tr>
<td></td>
<td></td>
<td>more accurate.</td>
</tr>
<tr>
<td>eClusteringAlg</td>
<td>ETcVnClusteringAlgorithm [141]</td>
<td>Clustering algorithm</td>
</tr>
<tr>
<td>nMaxClusterRad</td>
<td>DINT</td>
<td>Only used for the LBG clustering algorithm. Maximum allowed radius (&gt; 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>of a single cluster, i.e. clusters with a higher radius will be split into</td>
</tr>
<tr>
<td></td>
<td></td>
<td>smaller ones, until a global number of nDifferentColors is reached.</td>
</tr>
<tr>
<td>bSingleSplitSte</td>
<td>BOOL</td>
<td>Only used for the LBG clustering algorithm. If true, the global optimization</td>
</tr>
<tr>
<td>ps</td>
<td></td>
<td>is always run after a single cluster has been split. If false, several</td>
</tr>
<tr>
<td></td>
<td></td>
<td>clusters are split within the same step before applying the global</td>
</tr>
<tr>
<td></td>
<td></td>
<td>optimization. Applying the global optimization less often is faster, but can</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lead to less optimal results, especially having 2 nearby clusters that can</td>
</tr>
<tr>
<td></td>
<td></td>
<td>be represented by 1)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4042.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.15 **Image Filtering**

This group contains functions for filtering images.

**Functions**

**Blurring**
6.4.15.1  F_VN_BilateralFilter

Apply a Bilateral filter to smooth the image but preserve edges.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_BilateralFilter : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    nDiameter : DINT;
    fSigmaColor : LREAL;
    fSigmaSpace : LREAL;
    hrPrev : HRESULT;
```

F_VN_BilateralFilter

ipSrcImage  ITcVnImage
ipDestImage  Reference To ITcVnImage
nDiameter  DINT
fSigmaColor  LREAL
fSigmaSpace  LREAL
hrPrev  HRESULT
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ipSrcImage</code></td>
<td><code>ITcVnImage</code></td>
<td>Source image</td>
</tr>
<tr>
<td><code>ipDestImage</code></td>
<td>Reference To <code>ITcVnImage</code></td>
<td>Destination image (Must be different from <code>ipSrcImage</code>! An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td><code>nDiameter</code></td>
<td><code>DINT</code></td>
<td>Diameter of the pixel neighborhood used for filtering (&gt; 0). A larger value means that farther colors are mixed together.</td>
</tr>
<tr>
<td><code>fSigmaColor</code></td>
<td><code>LREAL</code></td>
<td>Sigma used for color space filtering (&gt; 0). A larger value means that farther colors are mixed together.</td>
</tr>
<tr>
<td><code>fSigmaSpace</code></td>
<td><code>LREAL</code></td>
<td>Sigma used for coordinate space filtering (&gt; 0). A larger value means that farther pixels can influence each other.</td>
</tr>
<tr>
<td><code>hrPrev</code></td>
<td><code>HRESULT</code></td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(<code>hrPrev</code>) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

`HRESULT` [135]

### Further information

The function `F_VN_BilateralFilter` applies a bilateral filter to the original image. The filter smooths the image and reduces noise while retaining the edges in the image.

### Parameter

**Original image**

The original image `ipSrcImage` must be of the element type `TCVN_ET_USINT` or a floating point type and have 1 or 3 channels.

**Result image**

The result image `ipDestImage` must be a different interface pointer to the original image `ipSrcImage`!

**Filter diameter**

The filter diameter `nDiameter` determines the size of a filter area. An odd number is expected. If `nDiameter = 0`, then the filter diameter is automatically calculated on the basis of `fSigmaSpace`.

**Color area**

This value may not be 0.

**Room area**

This value may not be 0.

### Expert parameters

Further parameters can be found in the expert version `F_VN_BilateralFilterExp` [1115].
Application

An application of this function can look like this, for example:

```c
hr := F_VN_BilateralFilter(
    ipSrcImage := ipImage,
    ipDestImage := ipImageRes,
    nDiameter := nDiameter,
    fSigmaColor := fSigmaColor,
    fSigmaSpace := fSigmaSpace,
    hrPrev := hr
);
```

The unprocessed original image (1st row) already exhibits fine structures on the surface of the gear wheels in the detail. In order to illustrate the effects of the filters, additional disturbances were added to the original image by Gauss noise (2nd row) and salt-and-pepper noise (3rd row).

<table>
<thead>
<tr>
<th>Original images</th>
<th>Result images:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Diameter = 7</td>
</tr>
<tr>
<td></td>
<td>Sigma = 100</td>
</tr>
</tbody>
</table>

![Original images](image1.png)
![Result images](image2.png)
Samples

- Blur filter [1415]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.15.2 F_VN_BilateralFilterExp

Apply a Bilateral filter to smooth the image but preserve edges. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```c
FUNCTION F_VN_BilateralFilterExp : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    nDiameter : DINT;
    fSigmaColor : LREAL;
    fSigmaSpace : LREAL;
    eBorderType : ETcVnBorderInterpolationMethod;
    hrPrev : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (Must be different from ipSrcImage! An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>nDiameter</td>
<td>DINT</td>
<td>Diameter of the pixel neighborhood used for filtering (e.g. 5, 7, 9). If &lt;= 0, it is automatically chosen dependend on fSigmaSpace.</td>
</tr>
<tr>
<td>fSigmaColor</td>
<td>LREAL</td>
<td>Sigma used for color space filtering (&gt; 0). A larger value means that farther colors are mixed together.</td>
</tr>
<tr>
<td>fSigmaSpace</td>
<td>LREAL</td>
<td>Sigma used for coordinate space filtering (&gt; 0). A larger value means that farther pixels can influence each other.</td>
</tr>
<tr>
<td>eBorderType</td>
<td>ETcVnBorderInterpolation Method [157]</td>
<td>Image border handling</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Further information

The function `F_VN_BilateralFilterExp` is the expert version of `F_VN_BilateralFilter` [1112]. It contains additional parameters.

### Parameter

#### Original image

The original image `ipSrcImage` must be of the element type TCVN_ET_USINT or a floating point type and have 1 or 3 channels.

#### Result image

The result image `ipDestImage` must be a different interface pointer to the original image `ipSrcImage`!

#### Filter diameter

The filter diameter `nDiameter` determines the size of a filter area. An odd number is expected. If `nDiameter = 0`, then the filter diameter is automatically calculated on the basis of `fSigmaSpace`.

#### Color area

This value may not be 0.

#### Room area

This value may not be 0.

#### Border extrapolation

The method of border extrapolation `eBorderType` defines how non-existent pixels are extrapolated beyond the image border in order to calculate the filter values at the image borders. For further details, see ETcVnBorderInterpolationMethod [157].
Application

An application of this function can look like this, for example:

```c
hr := F_VN_BilateralFilterExp(
   ipSrcImage := ipImage,
   ipDestImage := ipImageRes,
   nDiameter := nDiameter,
   fSigmaColor := fSigmaColor,
   fSigmaSpace := fSigmaSpace,
   eBorderType := eBorderType,
   hrPrev := hr
);
```

Samples

- Blur filter [1415]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.15.3 F_VN_BoxFilter

Apply a box filter to an image.

Syntax

Definition:

```c
FUNCTION F_VN_BoxFilter : HRESULT
VAR_INPUT
   ipSrcImage : ITcVnImage;
   ipDestImage : Reference To ITcVnImage;
   nFilterWidth : UDINT;
   nFilterHeight : UDINT;
   hrPrev : HRESULT;
END_VAR
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage[383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage[383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>nFilterWidth</td>
<td>UDINT</td>
<td>Filter width in pixels</td>
</tr>
<tr>
<td>nFilterHeight</td>
<td>UDINT</td>
<td>Filter height in pixels</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
### Return value

**HRESULT [135]**

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.15.4 F_VN_BoxFilterExp

**Applies a box filter to an image.** (expert function)

**Syntax**

**Definition:**

```c
FUNCTION F_VN_BoxFilterExp : HRESULT
VAR_INPUT
    ipSrcImage : ITCvnImage;
    ipDestImage : Reference To ITCvnImage;
    nFilterWidth : UDINT;
    nFilterHeight : UDINT;
    eDestDepth : ETCvnElementType;
    aAnchor : Reference To TcVnPoint;
    bNormalize : BOOL;
    eBorderType : ETCvnBorderInterpolationMethod;
    hrPrev : HRESULT;
END_VAR
```

---

**Function:**

- **F_VN_BoxFilterExp**
  - **ipSrcImage:** ITCvnImage
  - **ipDestImage:** Reference To ITCvnImage
  - **nFilterWidth:** UDINT
  - **nFilterHeight:** UDINT
  - **eDestDepth:** ETCvnElementType
  - **aAnchor:** Reference To TcVnPoint
  - **bNormalize:** BOOL
  - **eBorderType:** ETCvnBorderInterpolationMethod
  - **hrPrev:** HRESULT

**Description:**

- Applies a box filter to an image. (expert function)
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>nFilterWidth</td>
<td>UDINT</td>
<td>Filter width in pixels</td>
</tr>
<tr>
<td>nFilterHeight</td>
<td>UDINT</td>
<td>Filter height in pixels</td>
</tr>
<tr>
<td>eDestDepth</td>
<td>ETcVnElementType [186]</td>
<td>Destination image depth</td>
</tr>
<tr>
<td>aAnchor</td>
<td>Reference To TcVnP[nt[151]]</td>
<td>Anchor point of the kernel ([-1, -1] for center)</td>
</tr>
<tr>
<td>bNormalize</td>
<td>BOOL</td>
<td>If true, the kernel is normalized by nFilterWidth * nFilterHeight</td>
</tr>
<tr>
<td>eBorderType</td>
<td>ETcVnBorderInterpolation Method [157]</td>
<td>Image border handling</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.15.5 F_VN_BrightBorderObjects

Find bright objects connected to the image border within a gray-scale single-channel image.

**Syntax**

**Definition:**

FUNCTION F_VN_BrightBorderObjects : HRESULT
VAR_INPUT
   ipSrcImage    : ITcVnImage;
   ipDestImage   : Reference To ITcVnImage;
   hrPrev        : HRESULT;
END_VAR

F_VN_BrightBorderObjects
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ItcVnImage [383]</td>
<td>Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ItcVnImage [383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Further information

This function transfers all bright border objects in the input image to the result image.

## Algorithm

All objects in the image that are brighter than the surrounding pixels are removed.

## Parameter

Input image

The input image ipSrcImage may only have one channel and must have one of the following element types: USINT, UINT, INT, REAL or LREAL.

Result image

The result image ipDestImage returns the bright border objects.

## Application

```c
hr := F_VN_BrightBorderObjects(
    ipSrcImage := ipImageIn,
    ipDestImage := ipImageRes,
    hrPrev     := hr
);
```

## Related functions

- F_VN_DarkBorderObjects [1128]
• F_VN_FillHoles

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.15.6 F_VN_CreateStructuringElement

Creates a structuring element for the usage with morphological operators and allocate an appropriate data buffer. The initial reference count is set to one if a new image interface is created and kept, otherwise.

Syntax

Definition:

```vbnet
FUNCTION F_VN_CreateStructuringElement : HRESULT
VAR_INPUT
  ipStructuringElement : Reference To ITcVnImage;
  eShape               : ETcVnStructuringElementShape;
  nWidth               : UDINT;
  nHeight              : UDINT;
  hrPrev               : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipStructuringElement</td>
<td>Reference To ITcVnImage</td>
<td>Returns the created structuring element (Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>eShape</td>
<td>ETcVnStructuringElementShape</td>
<td>Shape of the structuring element (rectangle, cross, or ellipse)</td>
</tr>
<tr>
<td>nWidth</td>
<td>UDINT</td>
<td>Width</td>
</tr>
<tr>
<td>nHeight</td>
<td>UDINT</td>
<td>Height</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT

Further information

The function creates a structure element that can be used for morphological operations with F_VN_MorphologicalOperator.
Parameter

Structure element

The structure element created, `ipStructuringElement`, is returned as `ITcVnImage` by representing the structure element with pixels with the value `1`.

Shape

The shape `eShape` of the structure element is defined by the enum `ETcVnStructuringElementShape`. The following shapes are available:

![Shapes](image)

As the pixel values of the structure element are `1`, they were scaled up here for a better display.

Size

The size of the structure element is defined by the width `nWidth` and height `nHeight` of the image `ipStructuringElement` to be created.

Application

The creation of a square structure element of the size `9` looks like this, for example:

```c
hr := F_VN_CreateStructuringElement(
    ipStructuringElement := ipElement,
    eShape := TCVN_SES_RECTANGLE,
    nWidth := 9,
    nHeight := 9,
    hrPrev := hr
);
```

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT 3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.15.7 F_VN_CustomFilter

Apply a custom filter to the image.
Syntax

Definition:

FUNCTION F_VN_CustomFilter : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    eDestDepth : ETcVnElementType;
    stKernel : Reference To TcVnMatrix;
    hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>eDestDepth</td>
<td>ETcVnElementType [186]</td>
<td>Destination image depth</td>
</tr>
<tr>
<td>stKernel</td>
<td>Reference To TcVnMatrix [208]</td>
<td>Custom filter kernel with values of type REAL or LREAL</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Further information

The function F_VN_CustomFilter applies a user-defined filter to the input image. The filter kernel is defined by a matrix. This matrix is folded with the image.

Parameter

Input image
The input image ipSrcImage may have any available format.

Result image
The result image ipDestImage has the same format as the input image ipSrcImage.

Result depth
The result depth eDestDepth defines which element type the result image ipDestImage has. A larger element type can display more information.

Filter kernel
The user-defined filter kernel stKernel is transferred as a matrix. Permitted element types of the matrix are TCVN_ET_REAL and TCVN_ET_LREAL. A corresponding matrix of the type TcVnMatrix [208] can be created with the aid of the function F_VN_InitMatrixStruct [1272].

Expert parameters
The expert version F_VN_CustomFilterExp [1126] contains additional parameters.
Application

hr := F_VN_InitMatrixStruct(
    pSrcBuffer := ADR(aMatrixArray7x7),
    stDestMatrix := stKernelMatrix,
    nRows := 7,
    nCols := 7,
    eElementType := TCVN_ET_REAL,
    hrPrev := hr
);  
hr := F_VN_CustomFilter(
    ipSrcImage := ipImageIn,
    ipDestImage := ipImageRes,
    eDestDepth := TCVN_ET_USINT,
    stKernel := stKernelMatrix,
    hrPrev := hr
);

- The unprocessed original image (1st row) already exhibits fine structures on the surface of the gear wheels in the detail. In order to illustrate the effects of the filters, additional disturbances were added to the original image by Gauss noise (2nd row) and salt-and-pepper noise (3rd row). An average filter is implemented as an example; a more detailed description can be found in the sample for the Blur filter [1415]:
Samples
- Blur filter [1415]
- User-defined filters

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.15.8 F_VN_CustomFilterExp

Apply a custom filter to the image. (expert function)

Syntax

Definition:

```pascal
FUNCTION F_VN_CustomFilterExp : HRESULT
VAR_INPUT
    ipSrcImage : ITCvImage;
    ipDestImage : Reference To ITCvImage;
    eDestDepth : ETcVnElementType;
    stKernel : Reference To TcVnMatrix;
    aAnchor : Reference To TcVnPoint;
    fDelta : LREAL;
    eBorderType : ETcVnBorderInterpolationMethod;
    hrPrev : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITCvImage [383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>eDestDepth</td>
<td>ETcVnElementType [186]</td>
<td>Destination image depth</td>
</tr>
<tr>
<td>stKernel</td>
<td>Reference To TcVnMatrix [208]</td>
<td>Custom filter kernel with values of type REAL or LREAL</td>
</tr>
<tr>
<td>aAnchor</td>
<td>Reference To TcVnPoint [151]</td>
<td>Anchor point of the kernel</td>
</tr>
<tr>
<td>fDelta</td>
<td>LREAL</td>
<td>Value added to each pixel after filtering</td>
</tr>
<tr>
<td>eBorderType</td>
<td>ETcVnBorderInterpolationMethod [157]</td>
<td>Image border handling</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Further information

The function `F_VN_CustomFilterExp` is the expert version of `F_VN_CustomFilter [1122]`. It contains additional parameters.
Parameter

Input image
The input image ipSrcImage may have any available format.

Result image
The result image ipDestImage has the same format as the input image ipSrcImage.

Result depth
The result depth eDestDepth defines which element type the result image ipDestImage has. A larger element type can display more information.

Filter kernel
The user-defined filter kernel stKernel is transferred as a matrix. Permitted element types of the matrix are TCVN_ET_REAL and TCVN_ET_LREAL. A corresponding matrix of the type TcVnMatrix can be created with the aid of the function F_VN_InitMatrixStruct.

Anchor point
The anchor point aAnchor defines which point of the filter kernel is moved over each pixel of the input image ipSrcImage.

Delta
The constant delta fDelta is added to the result of the filter operation.

Border extrapolation
The method of border extrapolation eBorderType defines how non-existent pixels are extrapolated beyond the image border in order to calculate the filter values at the image borders. For further details, see ETcVnBorderInterpolationMethod.

Application

hr := F_VN_CustomFilterExp(
  ipSrcImage := ipImageIn,
  ipDestImage := ipImageRes,
  eDestDepth := TCVN_ET_USINT,
  stKernel := stKernelMatrix,
  aAnchor := aAnchorPoint,
  fDelta := 0,
  eBorderType := TCVN_BIM_DEFAULT,
  hrPrev := hr
);
6.4.15.9  F_VN_DarkBorderObjects

Find dark objects connected to the image border within a gray-scale single-channel image. (equivalent to filling holes)

Syntax

Definition:

```
FUNCTION F_VN_DarkBorderObjects : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

| HRESULT [135] |

Further information

This function transfers all dark border objects in the input image to the result image.

Algorithm

All objects in the image that are darker than the surrounding pixels are removed.

Parameter

Input image

The input image `ipSrcImage` may only have one channel and must have one of the following element types: USINT, UINT, INT, REAL or LREAL.

Result image

The result image `ipDestImage` returns the dark border objects.

Application

```
hr := F_VN_DarkBorderObjects(
    ipSrcImage := ipImageIn,
    ipDestImage := ipImageRes,
    hrPrev := hr
);
```
Input image

Result image

Related functions

- F_VN_FillHoles [1129] als Alias
- F_VN_BrightBorderObjects [1119]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.15.10   F_VN_FillHoles

```
FUNCTION F_VN_FillHoles : HRESULT
VAR_INPUT
  ipSrcImage : ITCvNImage;
  ipDestImage : Reference To ITCvNImage;
  hrPrev : HRESULT;
END_VAR
```

Fill holes within a gray-scale single-channel image. (equivalent to finding dark border objects)

Syntax

Definition:

```
FUNCTION F_VN_FillHoles : HRESULT
VAR_INPUT
  ipSrcImage : ITCvNImage;
  ipDestImage : Reference To ITCvNImage;
  hrPrev : HRESULT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ICTcVnImage</td>
<td>Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ICTcVnImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Further information**

This function fills holes in the input image.

**Algorithm**

All objects in the image that are darker than the surrounding pixels are removed.

**Parameter**

**Input image**

The input image ipSrcImage may only have one channel and must have one of the following element types: USINT, UINT, INT, REAL or LREAL.

**Result image**

The result image ipDestImage returns the image with the holes filled.

**Application**

hr := F_VN_FillHoles(ipSrcImage := ipImageIn, ipDestImage := ipImageRes, hrPrev := hr);

**Related functions**

- F_VN_DarkBorderObjects [1128] as alias
Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.15.11 F_VN_GaussianFilter

Apply a Gaussian filter to smooth the image.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```c
FUNCTION F_VN_GaussianFilter : HRESULT
VAR_INPUT
  ipSrcImage    : ITcVnImage;
  ipDestImage   : Reference To ITcVnImage;
  nFilterWidth  : UDINT;
  nFilterHeight : UDINT;
  hrPrev        : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>nFilterWidth</td>
<td>UDINT</td>
<td>Filter width in pixels (1, 3, 5, 7, ...)</td>
</tr>
<tr>
<td>nFilterHeight</td>
<td>UDINT</td>
<td>Filter height in pixels (1, 3, 5, 7, ...)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT

**Further information**

The function F_VN_GaussianFilter applies a Gaussian filter to the input image.
Algorithm

The Gaussian filter is often used in image processing to reduce noise or smooth an image. Each pixel is replaced by the weighted average of its environment, removing smaller details but preserving larger ones. The larger the filter, the more smoothed the image will be (and therefore appears more blurred).

Parameter

Input image

The input image `ipSrcImage` may have any available format.

Result image

The result image `ipDestImage` is given the same format as the input image `ipSrcImage`.

Filter size

The size of the Gauss filter is described in the X direction by `nFilterWidth` and in the Y direction by `nFilterHeight`. The size specifications must be odd numbers, as there must always be a central pixel.

Expert parameters

The expert version `F_VN_GaussianFilterExp` contains additional parameters.

Application

```c
hr := F_VN_GaussianFilter{
    ipSrcImage := ipImageIn,
    ipDestImage := ipImageRes,
    nFilterWidth := 7,
    nFilterHeight := 7,
    hrPrev := hr
};
```

The unprocessed original image (1st row) already exhibits fine structures on the surface of the gear wheels in the detail. In order to illustrate the effects of the filters, additional disturbances were added to the original image by Gauss noise (2nd row) and salt-and-pepper noise (3rd row).
Samples

- Blur filter [1415]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.15.12  F_VN_GaussianFilterExp

### Syntax

**Definition:**

FUNCTION F_VN_GaussianFilterExp : HRESULT
VAR_INPUT
  ipSrcImage : ITCvnImage;
  ipDestImage : Reference To ITCvnImage;
  nFilterWidth : UDINT;
  nFilterHeight : UDINT;
  fSigmaX : LREAL;
  fSigmaY : LREAL;
  eBorderType : ETcVnBorderInterpolationMethod;
  hrPrev : HRESULT;
END_VAR

**Return value**

HRESULT

**Further information**

The function **F_VN_GaussianFilterExp** is the expert version of **F_VN_GaussianFilter [1131]**. It contains additional parameters.

---

Apply a Gaussian filter to smooth the image. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvImage [1133]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITCvImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>nFilterWidth</td>
<td>UDINT</td>
<td>Filter width in pixels (1, 3, 5, 7, ...)</td>
</tr>
<tr>
<td>nFilterHeight</td>
<td>UDINT</td>
<td>Filter height in pixels (1, 3, 5, 7, ...)</td>
</tr>
<tr>
<td>fSigmaX</td>
<td>LREAL</td>
<td>Gaussian kernel standard deviation in X direction (&gt;= 0, automatically chosen if 0)</td>
</tr>
<tr>
<td>fSigmaY</td>
<td>LREAL</td>
<td>Gaussian kernel standard deviation in Y direction (&gt;= 0, automatically chosen if 0)</td>
</tr>
<tr>
<td>eBorderType</td>
<td>ETcVnBorderInterpolationMethod</td>
<td>Image border handling</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [1135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Parameter

Input image

The input image ipSrcImage may have any available format.

Result image

The result image ipDestImage is given the same format as the input image ipSrcImage.

Filter size

The size of the Gauss filter is described in the X direction by nFilterWidth and in the Y direction by nFilterHeight. The size specifications must be odd numbers, as there must always be a central pixel.

Standard deviation

The standard deviation in the X direction fSigmaX and in the Y direction fSigmaY defines not only the filter size but also the shape of the Gauss curve used in the filter.

Border extrapolation

The method of border extrapolation eBorderType defines how non-existent pixels are extrapolated beyond the image border in order to calculate the filter values at the image borders. For further details, see ETcVnBorderInterpolationMethod [157].

Application

hr := F_VN_GaussianFilterExp(
    ipSrcImage := ipImageIn,
    ipDestImage := ipImageRes,
    nFilterWidth := 3,
    nFilterHeight := 3,
    fSigmaX := 5,
    fSigmaY := 5,
    eBorderType := TCVN_BIM_DEFAULT,
    hrPrev := hr
);

Samples

- Blur filter [1415]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.15.13 F_VN_LaplacianFilter

Apply a Laplacian filter to an image.
Syntax

Definition:

```c
FUNCTION F_VN_LaplacianFilter : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    eDestDepth : ETcVnElementType;
    hrPrev : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>eDestDepth</td>
<td>ETcVnElementType [186]</td>
<td>Destination image depth</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Further information

The function `F_VN_LaplacianFilter` applies a Laplace filter to the input image.

Algorithm

The Laplace filter can be used to detect edges by approximating the second derivative of the transitions of adjacent pixel intensities. Since the derivative also has negative values, it is advisable to use a signed element type for the result image (e.g. TCVN_ET_INT).

For nKernelSize = 3, 5, ... the Laplace filter is calculated by adding the second derivative in x- and y-direction over the specified window size. For nKernelSize = 1, the following special 3 x 3 kernel is used:

```
[ 0  1  0 ]
[ 1 -4  1 ]
[ 0  1  0 ]
```

Parameter

Input image

The input image `ipSrcImage` may be in any available format, except for the data types SINT and DINT, which are generally not supported.

Result image

The result image `ipDestImage` has the same size as the input image `ipSrcImage`, but the pixel format is determined by the result depth `eDestDepth`.

Result depth

The result depth `eDestDepth` defines which element type the result image `ipDestImage` has. A larger element type can display more information.
**Combination of input image and result depth**

Only the same or a larger data type than that of the input image can be used as the result depth, otherwise data would be lost. For example, a UINT cannot be converted to a USINT. Furthermore, there are two restrictions, so that a UINT cannot be converted into an INT and a REAL cannot be converted into an LREAL.

**Expert parameters**

The expert version `F_VN_LaplacianFilterExp` contains additional parameters.

**Application**

The application of a Laplace filter with a result depth of 8 bits looks like this, for example:

```c
hr := F_VN_LaplacianFilter(
    ipSrcImage := ipImageIn,
    ipDestImage := ipImageRes,
    eDestDepth := TCVN_ET_USINT,
    hrPrev := hr
);
```

**Samples**

- Edge detection filters

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
Apply a Laplacian filter to an image. (expert function)

Syntax

Definition:

FUNCTION F_VN_LaplacianFilterExp : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    eDestDepth : ETcVnElementType;
    nKernelSize : UDINT;
    fScale : LREAL;
    fDelta : LREAL;
    eBorderType : ETcVnBorderInterpolationMethod;
    hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [1135]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>eDestDepth</td>
<td>ETcVnElementType [186]</td>
<td>Destination image depth</td>
</tr>
<tr>
<td>nKernelSize</td>
<td>UDINT</td>
<td>Aperture size used to compute the second-derivative filters (1, 3, 5, ..., 31)</td>
</tr>
<tr>
<td>fScale</td>
<td>LREAL</td>
<td>Scale factor for the computed derivative values</td>
</tr>
<tr>
<td>fDelta</td>
<td>LREAL</td>
<td>Delta value that is added to the results before storing them</td>
</tr>
<tr>
<td>eBorderType</td>
<td>ETcVnBorderInterpolationMethod [157]</td>
<td>Image border handling</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Further information

The function F_VN_LaplacianFilterExp is the expert version of F_VN_LaplacianFilter [1135]. It contains additional parameters.
The input image ipSrcImage may be in any available format, except for the data types SINT and DINT, which are generally not supported.

Result image

The result image ipDestImage has the same size as the input image ipSrcImage, but the pixel format is determined by the result depth eDestDepth.

Result depth

The result depth eDestDepth defines which element type the result image ipDestImage has. A larger element type can display more information.

### NOTE

**Combination of input image and result depth**

Only the same or a larger data type than that of the input image can be used as the result depth, otherwise data would be lost. For example, a UINT cannot be converted to a USINT. Furthermore, there are two restrictions, so that a UINT cannot be converted into an INT and a REAL cannot be converted into an LREAL.

Kernel size

The parameter nKernelSize defines the size of the filter kernel for the approximation of the 2nd derivative. Only odd numbers between 1 and 31 are allowed.

Scaling

The result of the Laplace operation is multiplied by the scaling factor fScale.

Delta

The constant delta fDelta is added to the result of the Scharr operation after scaling with fScale.

Border extrapolation

The method of border extrapolation eBorderType defines how non-existent pixels are extrapolated beyond the image border in order to calculate the filter values at the image borders. For further details, see ETCVnBorderInterpolationMethod [157].

Application

The application of a Laplace filter with a kernel size of $3 \times 3$ looks like this, for example:

```c
hr := F_VN_LaplacianFilterExp(
    ipSrcImage := ipImageIn,
    ipDestImage := ipImageRes,
    eDestDepth := TCVN_ET_USINT,
    nKernelSize := 3,
    fScale = 1,
    fDelta = 0,
    eBorderType := TCVN_BIM_DEFAULT,
    hrPrev := hr
);
```

Samples

- Edge detection filters [1417]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
Find local maxima in a gray-scale single-channel image. The found maxima are marked by a value of 1 in the destination image.

Syntax

Definition:

```c
FUNCTION F_VN_LocalMaxima : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  hrPrev : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image (USINT or UINT, 1 channel)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

```
HRESULT [135]
```

Further information

The function F_VN_LocalMaxima finds local maxima in the input image.

Parameter

Input image

The input image ipSrcImage must be a 1-channel grayscale image with a bit depth of 8 or 16 bits (USINT or UINT).

Result image

The result image ipDestImage is of the same type as the input image ipSrcImage. Each local maximum of the input image is marked here by a pixel value of 1.

Application

The finding of local maxima looks like this:

```c
hr := F_VN_LocalMaxima(
  ipSrcImage := ipImageIn,
  ipDestImage := ipImageRes,
  hrPrev := hr
);
```
Samples

- Removing local extremes

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.15.16 F_VN_LocalMinima

Find local minima in a gray-scale single-channel image. The found minima are marked by a value of 1 in the destination image.

Syntax

Definition:

```plaintext
FUNCTION F_VN_LocalMinima : HRESULT
VAR_INPUT
   ipSrcImage : ITcVnImage;
   ipDestImage : Reference To ITcVnImage;
   hrPrev : HRESULT;
END_VAR

RETURN HRESULT
```

## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image (USINT or UINT, 1 channel)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

Further information

The function F_VN_LocalMinima finds local minima in the input image.

Parameter

Input image

The input image ipSrcImage must be a 1-channel grayscale image with a bit depth of 8 or 16 bits (USINT or UINT).
Result image

The result image \textit{ipDestImage} is of the same type as the input image \textit{ipSrcImage}. Each local minimum of the input image is marked here by a pixel value of 1.

Application

The finding of local minima looks like this:

\begin{verbatim}
hr := F_VN_LocalMinima(
    ipSrcImage := ipImageIn,
    ipDestImage := ipImageRes,
    hrPrev := hr
);
\end{verbatim}

Samples

- Removing local extremes

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.15.17 \textbf{F_VN_MedianFilter}

\textbf{F_VN_MedianFilter}

\begin{itemize}
  \item ipSrcImage \textit{ITcVnImage}
  \item ipDestImage \textit{Reference To ITcVnImage}
  \item nFilterSize \textit{UDINT}
  \item hrPrev \textit{HRESULT}
\end{itemize}

Apply a Median filter to an image.

Syntax

\textbf{Definition:}

\texttt{FUNCTION F_VN_MedianFilter : HRESULT}
\texttt{VAR_INPUT}
\texttt{    ipSrcImage : ITcVnImage;}
\texttt{    ipDestImage : Reference To ITcVnImage;}
\texttt{    nFilterSize : UDINT;}
\texttt{    hrPrev : HRESULT;}
\texttt{END_VAR}

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>\textit{ITcVnImage} [\ref{383}]</td>
<td>Source image (for nFilterSize 3 or 5: USINT, UINT, REAL. For bigger filters, only USINT is supported.)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>\textit{Reference To ITcVnImage} [\ref{383}]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
</tbody>
</table>
| nFilterSize| UDINT        | Size (width and height) of the filter (3, 5, 7, ...)
| hrPrev     | \textit{HRESULT} [\ref{135}] | HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.) |
Return value

HRESULT \[135\]

Further information

The function F_VN_MedianFilter applies a median filter to the input image.

Algorithm

The median filter is a ranking filter that is often used to suppress noise. All pixel values captured by the filter matrix are sorted by size, then the current value is replaced by the value at the center position of the sorted list. The advantage over a filter with a weighted mean value (e.g. a Gaussian filter) is that strong outliers of individual pixels (such as the 155 in the following sample image) are completely ignored and thus do not have a negative effect on the result.

The size of the filter matrix is specified by \(nFilterSize\), only odd integers \(\geq 3\) being accepted.

Parameter

Input image

The input image \(ipSrcImage\) can have any number of channels. The element types \(USINT\); \(UINT\) and \(REAL\) are permitted with a filter size \(nFilterSize\) of 3 or 5. Only the element type \(USINT\) is supported for larger filters.

Result image

The result image \(ipDestImage\) returns the filtered image.

Filter size

The filter size \(nFilterSize\) defines the height and width of the filter area. It must be an odd number and \(\geq 3\).

Application

The application of a median filter with the size \(7\times7\) looks like this, for example:

```c
hr := F_VN_MedianFilter(
    ipSrcImage := ipImageIn,
    ipDestImage := ipImageRes,
    nFilterSize := 7,
    hrPrev := hr
);```

The unprocessed original image (1st row) already exhibits fine structures on the surface of the gear wheels in the detail. In order to illustrate the effects of the filters, additional disturbances were added to the original image by Gauss noise (2nd row) and salt-and-pepper noise (3rd row).
Samples

- Blur filter [1415]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
Apply a morphological operator.

Syntax

Definition:

```sql
FUNCTION F_VN_MorphologicalOperator : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    eOperator : ETcVnMorphologicalOperator;
    ipStructuringElement : ITcVnImage;
    hrPrev : HRESULT;
END_VAR
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>eOperator</td>
<td>ETcVnMorphologicalOperator</td>
<td>Operator type</td>
</tr>
<tr>
<td>ipStructuringElement</td>
<td>ITcVnImage</td>
<td>Structuring element to be applied (Typically created via F_VN_CreateStructuringElement.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

```sql
HRESULT [135]
```

Further information

The function F_VN_MorphologicalOperator applies a morphological operator to the input image.

Parameter

Original image

There are no special requirements for the original image ipSrcImage.

Result image

The modified image is returned in the result image ipDestImage.

Morphological operator

The parameter eOperator defines which morphological operator is applied to the original image ipSrcImage. All operators in the enum ETcVnMorphologicalOperator [195] are available.
Structure element

The structure element `ipStructuringElement` is the element that specifies the filter area of the morphological operator. It can be created with the help of the function `F_VN_CreateStructuringElement` [1121].

Application

The application of an Opening operator with a rectangular filter area of the size $3 \times 3$ looks like this, for example:

```c
VAR
  ipElement : ITcVnImage;
END_VAR

hr := F_VN_CreateStructuringElement(ipElement, TCVN_SES_RECTANGLE, 3, 3, hr);

hr := F_VN_MorphologicalOperator(
  ipSrcImage := ipImageIn,
  ipDestImage := ipImageRes,
  eOperator := TCVN_MO_OPENING,
  ipStructuringElement := ipElement,
  hrPrev := hr
);
```

Samples

- Morphological operators [1419]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.15.19 F_VN_RemoveLocalMaxima

```c
FUNCTION F_VN_RemoveLocalMaxima : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  fHeight : LREAL;
  hrPrev : HRESULT;
END_VAR

F_VN_RemoveLocalMaxima
```

Remove local maxima up to a given height from a gray-scale single-channel image.

Syntax

Definition:

FUNCTION F_VN_RemoveLocalMaxima : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  fHeight : LREAL;
  hrPrev : HRESULT;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage[称之为383]</td>
<td>Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>fHeight</td>
<td>LREAL</td>
<td>Maximum height of the maxima to be removed (must be greater than 0)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[称之为135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT[称之为135]

## Further information

The function F_VN_RemoveLocalMaxima removes local maxima from the input image.

### Parameter

#### Input image

The input image ipSrcImage may only have one channel and must have one of the following element types: USINT, UINT, INT, REAL or LREAL.

#### Result image

The result image ipDestImage returns the image from which the local maxima have been removed.

#### Maximum height

The maximum height fHeight defines the height up to which maxima in the input image are removed. The height describes the difference in intensity between the maximum and the value to which the pixels concerned are set. fHeight must be larger than 0.

### Application

The removal of local maxima up to a height of 10 looks like this, for example:

```c
hr := F_VN_RemoveLocalMaxima(  
ipSrcImage := ipImageIn,  
ipDestImage := ipImageRes,  
fHeight := 10,  
hrPrev := hr)
```

### Samples

- Removing local extremes

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.15.20 F_VN_RemoveLocalMinima

Remove local minima up to a given height from a gray-scale single-channel image.

Syntax

Definition:

FUNCTION F_VN_RemoveLocalMinima : HRESULT
VAR_INPUT
  ipSrcImage        : ITcVnImage;
  ipDestImage       : Reference To ITcVnImage;
  fHeight           : LREAL;
  hrPrev            : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage[383]</td>
<td>Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>fHeight</td>
<td>LREAL</td>
<td>Maximum height of the minima to be removed (must be greater than 0)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT[135]

Further information

The function F_VN_RemoveLocalMinima removes local minima from the input image.

Parameter

Input image

The input image ipSrcImage may only have one channel and must have one of the following element types: USINT, UINT, INT, REAL or LREAL.

Result image

The result image ipDestImage returns the image from which the local minima have been removed.

Maximum height

The maximum height fHeight defines the height up to which minima in the input image are removed. The height describes the difference in intensity between the minimum and the value to which the pixels concerned are set. fHeight must be larger than 0.
Application

The removal of local minima up to a height of 10 looks like this, for example:

```c
hr := F_VN_RemoveLocalMinima(
    ipSrcImage := ipImageIn,
    ipDestImage := ipImageRes,
    fHeight := 10,
    hrPrev := hr
);
```

Samples

- Removing local extremes

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.15.21 F_VN_ScharrFilter

Calculates the first order derivative in x or y direction using a Scharr filter.

Syntax

Definition:

```c
FUNCTION F_VN_ScharrFilter : HRESULT
VAR_INPUT
    ipSrcImage : ITCvImage;
    ipDestImage : Reference To ITCvImage;
    eDestDepth : ETCvElementType;
    eFilterDirection : ETCvFilterDirection;
    hrPrev : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITCvImage [383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>eDestDepth</td>
<td>ETCvElementType [186]</td>
<td>Destination image depth</td>
</tr>
<tr>
<td>eFilterDirection</td>
<td>ETCvFilterDirection [189]</td>
<td>Filter direction</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Further information

The function F_VN_ScharrFilter applies a Scharr filter to the input image. The Scharr filter functions analogously to the Sobel Filter [1157] and differs from it only by the kernel.

Algorithm

The Scharr filter is closely related to the Sobel filter. However, it provides more accurate results than a 3 x 3 Sobel filter.

\[
G_x = \begin{bmatrix}
-3 & 0 & +3 \\
-10 & 0 & +10 \\
-3 & 0 & +3 \\
\end{bmatrix} \quad G_y = \begin{bmatrix}
-3 & -10 & -3 \\
0 & 0 & 0 \\
+3 & +10 & +3 \\
\end{bmatrix}
\]

Parameter

Original image

The original image ipSrcImage can have any format.

Result image

The result image ipDestImage returns the filter result and has the same format as the original image ipSrcImage.

Result depth

The result depth eDestDepth defines which element type the result image ipDestImage has. A larger element type can display more information.

Filter direction

The filter direction eFilterDirection defines whether the Scharr filter is applied to the image in the X-direction (TCVN_FD_X) or in the Y-direction (TCVN_FD_Y).

Expert parameters

Further parameters can be found in the expert version F_VN_ScharrFilterExp [1151].

Application

```c
hr := F_VN_ScharrFilter(
ipSrcImage := ipImageIn,
ipDestImage := ipImageRes,
eDestDepth := TCVN_ET_USINT,
eFilterDirection:= TCVN_FD_X,
hrPrev := hr,
);
```
Samples

- Edge detection filters [1417]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.1522  F_VN_ScharrFilterExp

Calculates the first order derivative in x or y direction using a Scharr filter. (expert function)

Syntax

**Definition:**

```
FUNCTION F_VN_ScharrFilterExp : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    eDestDepth : ETcVnElementType;
    eFilterDirection : ETcVnFilterDirection;
    fScale : LREAL;
    fDelta : LREAL;
    eBorderType : ETcVnBorderInterpolationMethod;
    hrPrev : HRESULT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [{383}]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [{383}]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>eDestDepth</td>
<td>ETcVnElementType [{186}]</td>
<td>Destination image depth</td>
</tr>
<tr>
<td>eFilterDirection</td>
<td>ETcVnFilterDirection [{189}]</td>
<td>Filter direction</td>
</tr>
<tr>
<td>fScale</td>
<td>LREAL</td>
<td>Scale factor for the computed derivative values</td>
</tr>
<tr>
<td>fDelta</td>
<td>LREAL</td>
<td>Delta value that is added to the results prior to storing them in dest</td>
</tr>
<tr>
<td>eBorderType</td>
<td>ETcVnBorderInterpolation Method [{157}]</td>
<td>Image border handling</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [{135}]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [{135}]

**Further information**

The function F_VN_ScharrFilterExp is the expert version of F_VN_ScharrFilter [{1149}]. It contains additional parameters.

**Parameter**

**Original image**

The original image ipSrcImage can have any format.

**Result image**

The result image ipDestImage returns the filter result and has the same format as the original image ipSrcImage.

**Result depth**

The result depth eDestDepth defines which element type the result image ipDestImage has. A larger element type can display more information.

**Filter direction**

The filter direction eFilterDirection defines whether the Scharr filter is applied to the image in the X-direction (TCVN_FD_X) or in the Y-direction (TCVN_FD_Y).

**Scaling**

The result of the Scharr operation is multiplied by the scaling factor fScale.

**Delta**

The constant delta fDelta is added to the result of the Scharr operation after scaling with fScale.

**Border extrapolation**
The method of border extrapolation eBorderType defines how non-existent pixels are extrapolated beyond the image border in order to calculate the filter values at the image borders. For further details, see ETcVnBorderInterpolationMethod [157].

Application

```c
hr := F_VN_ScharrFilterExp(
    ipSrcImage := ipImageIn,
    ipDestImage := ipImageRes,
    eDestDepth := TCVN_ET_USINT,
    eFilterDirection := TCVN_FD_X,
    fScale := 1,
    fDelta := 0,
    eBorderType := TCVN_BIM_DEFAULT,
    hrPrev := hr,
);
```

Samples

- Edge detection filters [1417]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.15.23 F_VN_SeparableCustomFilter

Apply a separable custom filter to the image.

Syntax

```c
FUNCTION F_VN_SeparableCustomFilter : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    eDestDepth : ETcVnElementType;
    stKernelX : Reference To TcVnMatrix;
    stKernelY : Reference To TcVnMatrix;
    hrPrev : HRESULT;
END_VAR
```
API reference

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage[383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>eDestDepth</td>
<td>ETcVnElementType [186]</td>
<td>Destination image depth</td>
</tr>
<tr>
<td>stKernelX</td>
<td>Reference To TcVnMatrix [208]</td>
<td>1D custom row-filter kernel with values of type REAL or LREAL</td>
</tr>
<tr>
<td>stKernelY</td>
<td>Reference To TcVnMatrix [208]</td>
<td>1D custom column-filter kernel with values of type REAL or LREAL</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Further information**

The function F_VN_SeparableCustomFilter applies a separable 2D filter to the input image.

**Algorithm**

A separable 2D filter can be combined from two 1D filters. It is more efficient to apply a 1D filter to an image twice than to apply a 2D filter once. For example, the well-known Sobel filter [1157] can be constructed as follows from two separate one-dimensional filters:

\[
stKernelX = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}, \quad stKernelY = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}
\]

\[
G = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 2 & 2 \\ -1 & 0 & 1 \end{bmatrix}
\]

**Parameter**

**Input image**

The input image ipSrcImage may have any available format.

**Result image**

The result image ipDestImage has the same size as the input image ipSrcImage, but the pixel format is determined by the result depth eDestDepth.

**X and Y kernel**

The two 1-dimensional filter kernels stKernelX and stKernelY are transferred as a matrix. Permitted element types of the matrix are TCVN_ET_REAL and TCVN_ET_LREAL. Corresponding matrices of the type TcVnMatrix [208] can be created with the aid of the function F_VN_InitMatrixStruct [1272].

**Expert parameters**

The expert version F_VN_SeparableCustomFilterExp [1155] contains additional parameters.
Application

The creation and application of a separable 2D filter looks like this, for example:

```c
hr := F_VN_SeparableCustomFilter(
    ipSrcImage := ipImageIn,
    ipDestImage := ipImageRes,
    eDestDepth := TCVN_ET_USINT,
    stKernelX := stKernelMatrixX,
    stKernelY := stKernelMatrixY,
    hrPrev := hr
);
```

Samples

- User-defined filters

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.1524 F_VN_SeparableCustomFilterExp

Apply a separable custom filter to the image. (expert function)

Syntax

Definition:

```c
FUNCTION F_VN_SeparableCustomFilterExp : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    eDestDepth : TCVNElementType;
    stKernelX : Reference To TcVnMatrix;
    stKernelY : Reference To TcVnMatrix;
    aAnchor : Reference To TcVnPoint;
    fDelta : LREAL;
    eBorderType : ETcVnBorderInterpolationMethod;
    hrPrev : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage[383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage[383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>eDestDepth</td>
<td>ETcVnElementType[186]</td>
<td>Destination image depth</td>
</tr>
<tr>
<td>stKernelX</td>
<td>Reference To TcVnMatrix[208]</td>
<td>1D custom row-filter kernel with values of type REAL or LREAL</td>
</tr>
<tr>
<td>stKernelY</td>
<td>Reference To TcVnMatrix[208]</td>
<td>1D custom column-filter kernel with values of type REAL or LREAL</td>
</tr>
<tr>
<td>aAnchor</td>
<td>Reference To TcVnPoint[151]</td>
<td>Anchor point of the kernel</td>
</tr>
<tr>
<td>fDelta</td>
<td>LREAL</td>
<td>Value added to each pixel after filtering</td>
</tr>
<tr>
<td>eBorderType</td>
<td>ETcVnBorderInterpolationMethod[157]</td>
<td>Image border handling</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT[135]

## Further information

The function F_VN_SeparableCustomFilterExp is the expert version of F_VN_SeparableCustomFilter[1153]. It contains additional parameters.

## Parameter

**Input image**

The input image ipSrcImage may have any available format.

**Result image**

The result image ipDestImage has the same size as the input image ipSrcImage, but the pixel format is determined by the result depth eDestDepth.

**X and Y kernel**

The two 1-dimensional filter kernels stKernelX and stKernelY are transferred as a matrix. Permitted element types of the matrix are TCVN_ET_REAL and TCVN_ET_LREAL. Corresponding matrices of the type TcVnMatrix[208] can be created with the aid of the function F_VN_InitMatrixStruct[1272].

**Anchor point**

The anchor point aAnchor defines which point of the filter kernel is moved over each pixel of the input image ipSrcImage.

**Delta**

The constant delta fDelta is added to the result of the filter operation.

**Border extrapolation**
The method of border extrapolation \texttt{eBorderType} defines how non-existent pixels are extrapolated beyond the image border in order to calculate the filter values at the image borders. For further details, see \texttt{ETcVnBorderInterpolationMethod} [\textsuperscript{157}].

**Application**

```plaintext
hr := F_VN_SeparableCustomFilterExp(
    ipSrcImage := ipImageIn,
    ipDestImage := ipImageRes,
    eDestDepth := TCVN_ET_USINT,
    stKernelX := stKernelMatrixX,
    stKernelY := stKernelMatrixY,
    aAnchor := aAnchorPoint,
    fDelta := 0,
    eBorderType := TCVN_BIM_DEFAULT,
    hrPrev := hr
);
```

**Samples**

- User-defined filters

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.15.25 F_VN_SobelFilter**

Calculates the first, second, or mixed image derivatives using an extended Sobel filter.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_SobelFilter : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    eDestDepth : ETcVnElementType;
    nXOrder : UDINT;
    nYOrder : UDINT;
    hrPrev : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage <img src="60x766" alt="383" /></td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage <img src="60x766" alt="383" /></td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>eDestDepth</td>
<td>ETcVnElementType <img src="57x785" alt="186" /></td>
<td>Destination image depth</td>
</tr>
<tr>
<td>nXOrder</td>
<td>UDINT</td>
<td>Order of the x-derivative (0, 1, 2)</td>
</tr>
<tr>
<td>nYOrder</td>
<td>UDINT</td>
<td>Order of the y-derivative (0, 1, 2)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT <img src="57x785" alt="135" /></td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT ![135](57x785)

## Further information

The function `F_VN_SobelFilter` applies a Sobel filter to the input image. The Sobel filter is used for edge detection. Edges in the input image are highlighted as bright lines on a dark background in the result image.

### Algorithm

The extended Sobel filter can be used to detect edges by approximating the nth derivative of the transitions of adjacent pixel intensities. Since the derivative also has negative values, it is advisable to use a signed element type for the result image (e.g. `TCVN_ET_INT`).

The kernel of the filter is determined by the parameters of the function. Typically the function is called with nXOrder = 1, nYOrder = 0, nKernelSize = 3 or nXOrder = 0, nYOrder = 1, nKernelSize = 3. These parameters lead to the kernels $G_x$ and $G_y$.

$$
G_x = \begin{bmatrix}
-1 & 0 & 1 \\
-2 & 0 & 2 \\
-1 & 0 & 1 \\
\end{bmatrix} \\
G_y = \begin{bmatrix}
1 & 2 & 1 \\
0 & 0 & 0 \\
-1 & -2 & -1 \\
\end{bmatrix}
$$

For these kernels, the filter combines Gaussian smoothing with the calculation of the first derivative, so that the result is quite noise resistant. $G_x$ emphasizes vertical edges, $G_y$ emphasizes horizontal edges.

### Parameter

**Original image**

The original image `ipSrcImage` can have any format.

**Result image**

The result image `ipDestImage` returns the filter result and has the same format as the original image `ipSrcImage`.

**Result depth**

The result depth `eDestDepth` defines which element type the result image `ipDestImage` has. A larger element type can display more information.

**Derivative order**

The derivative orders `nXOrder` and `nYOrder` define which derivative in the X and Y direction is to be used to create the filter.
Expert parameters

Further parameters can be found in the expert version F_VN_SobelFilterExp [1159].

Application

```c
hr := F_VN_SobelFilter(
    ipSrcImage := ipImageIn,
    ipDestImage := ipImageRes,
    eDestDepth := TCVN_ET_USINT,
    nXOrder := 1,
    nYOrder := 1,
    hrPrev := hr,
);```

Samples

- Edge detection filters [1417]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.15.26 F_VN_SobelFilterExp

![Original image](image1) ![Result image](image2)
Calculates the first, second, third, or mixed image derivatives using an extended Sobel filter. (expert function)

**Syntax**

**Definition:**

FUNCTION F_VN_SobelFilterExp : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  eDestDepth : ETcVnElementType;
  nXOrder : UDINT;
  nYOrder : UDINT;
  nKernelSize : UDINT;
  fScale : LREAL;
  fDelta : LREAL;
  eBorderType : ETcVnBorderInterpolationMethod;
  hrPrev : HRESULT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [1383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [1383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>eDestDepth</td>
<td>ETcVnElementType [186]</td>
<td>Destination image depth</td>
</tr>
<tr>
<td>nXOrder</td>
<td>UDINT</td>
<td>Order of the x-derivative (must be &lt; nKernelSize)</td>
</tr>
<tr>
<td>nYOrder</td>
<td>UDINT</td>
<td>Order of the y-derivative (must be &lt; nKernelSize)</td>
</tr>
<tr>
<td>nKernelSize</td>
<td>UDINT</td>
<td>Size of the extended Sobel kernel (3, 5, 7, ..., 31)</td>
</tr>
<tr>
<td>fScale</td>
<td>LREAL</td>
<td>Scale factor for the computed derivative values</td>
</tr>
<tr>
<td>fDelta</td>
<td>LREAL</td>
<td>Delta value that is added to the results prior to storing them in dest</td>
</tr>
<tr>
<td>eBorderType</td>
<td>ETcVnBorderInterpolationMethod [157]</td>
<td>Image border handling</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Further information**

The function F_VN_SobelFilterExp is the expert version of F_VN_SobelFilter [1157]. It contains additional parameters.

**Parameter**

**Original image**

The original image ipSrcImage can have any format.

**Result image**

The result image ipDestImage returns the filter result and has the same format as the original image ipSrcImage.

**Result depth**
The result depth \( e_{DestDepth} \) defines which element type the result image \( ip_{DestImage} \) has. A larger element type can display more information.

**Derivative order**

The derivative orders \( n_{XOrder} \) and \( n_{YOrder} \) define which derivative in the X and Y direction is to be used to create the filter.

**Kernel size**

The size \( n_{KernelSize} \) of the Sobel kernel determines how large the area is that influences the result value of a pixel.

**Scaling**

The result of the Sobel operation is multiplied by the scaling factor \( f_{Scale} \).

**Delta**

The constant delta \( f_{Delta} \) is added to the result of the Sobel operation after scaling with \( f_{Scale} \).

**Border extrapolation**

The method of border extrapolation \( e_{BorderType} \) defines how non-existent pixels are extrapolated beyond the image border in order to calculate the filter values at the image borders. For further details, see \( ETcVnBorderInterpolationMethod \).

**Application**

```c
hr := F_VN_SobelFilterExp(
    ipSrcImage := ipImageIn,
    ipDestImage := ipImageRes,
    eDestDepth := TCVN_ET_USINT,
    nXOrder := 1,
    nYOrder := 1,
    nKernelSize := 3,
    fScale := 1,
    fDelta := 0,
    eBorderType := TCVN_BIM_DEFAULT,
    hrPrev := hr,
);
```

**Samples**

- Edge detection filters [1417]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.16 Image Segmentation

This group contains functions for segmenting images.

**Functions**

**Segmentation**

- F_VN_AdaptiveThreshold(Exp) [1163]
**F_VN_AdaptiveThreshold**

Apply an adaptive threshold to a gray level image. Can return partial results when canceled by Watchdog.

**Syntax**

**Definition:**

```c
FUNCTION F_VN_AdaptiveThreshold : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    fMaxValue : LREAL;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image (1 channel, elements of type USINT)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>fMaxValue</td>
<td>LREAL</td>
<td>Value assigned to pixels for which the threshold condition is true</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Further information**

The function F_VN_AdaptiveThreshold applies an adaptive threshold to the input image.

**Parameter**

**Input image**

The input image ipSrcImage must be a 1-channel grayscale image with a bit depth of 8 bits (USINT).
Result image

The binary result image \(\text{ipDestImage}\) has the same format as the input image. The pixels have either the value 0 or the target value \(f\text{MaxValue}\).

Target value

The target value \(f\text{MaxValue}\) defines which value should be given to the pixels that fulfil the threshold criterion. If the result image is to consist only of black and white pixels, then the maximum value of the element type of the input image must be set here. This corresponds to \((2^d)-1\), where \(d\) represents the bit depth of the element type; e.g. \((2^8)-1=255\) with 8-bit images.

Expert parameters

The expert version \(\text{F\_VN\_AdaptiveThresholdExp}[\cdot 1163]\) contains additional parameters.

Application

```c
hr := \text{F\_VN\_AdaptiveThresholdExp}( \\
  \text{ipSrcImage} := \text{ipImageIn}, \\
  \text{ipDestImage} := \text{ipImageRes}, \\
  \text{fMaxValue} := 255, // max value of 8-bit image: (2^8)-1=255 \\
  \text{hrPrev} := \text{hr} \\
);
```

Related functions

- \(\text{F\_VN\_Threshold}[\cdot 1166]\) for binary segmentation on the basis of a threshold value
- \(\text{F\_VN\_AdaptiveThreshold}[\cdot .1162]\) for binary, adaptive segmentation
- \(\text{F\_VN\_CheckColorRange}[\cdot 1165]\) for binary segmentation on the basis of a color spectrum
- \(\text{F\_VN\_ReferenceColorSimilarity}[\cdot 1097]\) for binary segmentation on the basis of a color model

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.16.2 \(\text{F\_VN\_AdaptiveThresholdExp}\)

Apply an adaptive threshold to a grayscale image. (expert function)
Can return partial results when canceled by Watchdog.

Syntax

Definition:
FUNCTION F_VN_AdaptiveThresholdExp : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  ipDestImage : Reference To ITcVnImage;
  fMaxValue : LREAL;
  eAdaptiveMethod : ETcVnAdaptiveThresholdMethod;
  eThresholdType : ETcVnThresholdType;
  nBlockSize : UDINT;
  fConstant : LREAL;
  hrPrev : HRESULT;
END_VAR

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image (1 channel, elements of type USINT)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>fMaxValue</td>
<td>LREAL</td>
<td>Value assigned to pixels for which the threshold condition is true</td>
</tr>
<tr>
<td>eAdaptiveMethod</td>
<td>ETcVnAdaptiveThresholdMethod</td>
<td>Adaptive threshold method to be applied (MEAN: threshold value will be calculated as a mean of the nBlockSize x nBlockSize neighborhood of (x,y) minus fConstant. GAUSSIAN: threshold value is a weighted sum (cross-correlation with a Gaussian window) of the nBlockSize x nBlockSize neighborhood of (x,y) minus fConstant.</td>
</tr>
<tr>
<td>eThresholdType</td>
<td>ETcVnThresholdType</td>
<td>Threshold type to be applied (only BINARY and BINARY_INV are supported)</td>
</tr>
<tr>
<td>nBlockSize</td>
<td>UDINT</td>
<td>Size of the pixel neighborhood to calculate the local threshold (3, 5, 7, ...)</td>
</tr>
<tr>
<td>fConstant</td>
<td>LREAL</td>
<td>Constant that is subtracted from the weighted mean of the pixel neighborhood, which leads to the local threshold</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Further information

The function F_VN_AdaptiveThresholdExp is the expert version of F_VN_AdaptiveThreshold [1162]. It contains additional parameters.

### Parameter

### Input image

The input image ipSrcImage must be a 1-channel grayscale image with a bit depth of 8 bits (USINT).

### Result image

The binary result image ipDestImage has the same format as the input image. The pixels have either the value 0 or the target value fMaxValue.
Target value

The target value fMaxValue defines which value should be given to the pixels that fulfil the threshold criterion. If the result image is to consist only of black and white pixels, then the maximum value of the element type of the input image must be set here. This corresponds to \((2^d)-1\), where \(d\) represents the bit depth of the element type; e.g. \((2^8)-1=255\) with 8-bit images.

Threshold method

The threshold method eAdaptiveMethod defines whether the adaptive threshold is calculated as an average or as a Gaussian weighted sum. The two values TCN_ATM_MEAN and TCN_ATM_GAUSSIAN are available for this in the enum ETcVnAdaptiveThresholdMethod [154].

Threshold inversion

The parameter eThresholdType specifies whether or not the result image should be inverted. Therefore, only the two values TCN_TT_BINARY and TCN_TT_BINARY_INV from the enum ETcVnThresholdType [201] are supported at this point.

Threshold neighborhood size

The parameter nBlockSize defines the size of the neighborhood that is considered for the calculation of the local threshold. The parameter must be an odd number (3, 5, 7, etc.).

Threshold adjustment

The constant value fConstant is subtracted from each local average value. The result is the local threshold that is used for the calculation of the result image.

Application

```c
hr := F_VN_AdaptiveThreshold(
    ipSrcImage := ipImageIn,
    ipDestImage := ipImageRes,
    fMaxValue := 255, // max value of 8-bit image: (2^8)-1=255
    eAdaptiveMethod := TCN_ATM_MEAN,
    eThresholdType := TCN_TT_BINARY_INV,
    nBlockSize := 5,
    fConstant := 10,
    hrPrev := hr
);
```

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.16.3 F_VN_CheckColorRange
Check if the pixel values of an image lie in a given range. The destination image has the same size as the source image but only one channel with a pixel element size of 8 bit. Its elements are set to 255 if the corresponding pixels of the source image are in the checked range and set to 0 otherwise.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_CheckColorRange : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    aLowerBounds : Reference To TcVnVector4_LREAL;
    aUpperBounds : Reference To TcVnVector4_LREAL;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage[383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage[383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>aLowerBounds</td>
<td>Reference To TcVnVector4_LREAL[153]</td>
<td>Channel-wise lower bounds (Unused channels are ignored.)</td>
</tr>
<tr>
<td>aUpperBounds</td>
<td>Reference To TcVnVector4_LREAL[153]</td>
<td>Channel-wise upper bounds (Unused channels are ignored.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT[135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.16.4 F_VN_Threshold**

```plaintext
F_VN_Threshold
    ipSrcImage ITcVnImage                      HRESULT F_VN_Threshold
    ipDestImage Reference To ITcVnImage
    fThreshold LREAL
    fMaxValue LREAL
    eThresholdType ETCvnThresholdType
    hrPrev HRESULT
```
Apply a fixed threshold or a dynamic threshold according to Otsu. Can use available TwinCAT Job Tasks for executing parallel code regions. Can return partial results when canceled by Watchdog.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_Threshold : HRESULT
VAR_INPUT
  ipSrcImage     : ITcVnImage;
  ipDestImage    : Reference To ITcVnImage;
  fThreshold     : LREAL;
  fMaxValue      : LREAL;
  eThresholdType : ETcVnThresholdType;
  hrPrev         : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image (For Otsu and Triangle threshold types, only 1 channel of TCVN_ET_USINT is supported)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Destination image (An appropriate destination image will be created if required.)</td>
</tr>
<tr>
<td>fThreshold</td>
<td>LREAL</td>
<td>Fixed threshold (unused if dynamic thresholding is selected)</td>
</tr>
<tr>
<td>fMaxValue</td>
<td>LREAL</td>
<td>Maximum pixel value</td>
</tr>
<tr>
<td>eThresholdType</td>
<td>ETcVnThresholdType [201]</td>
<td>Threshold type to be applied</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Further information**

The function `F_VN_Threshold` segments the input image with regard to a threshold. Depending on this threshold and the algorithm used, each pixel of the binary result image is assigned the value 0 and a specified target value.

**Algorithm**

A threshold transforms a monochrome image into a binary image and thus creates segments. Good segmentation is a prerequisite for many subsequent methods such as contour search. TwinCAT Vision offers different types of threshold values that are suitable for different applications.

**Fixed threshold value**

A fixed threshold simply creates a binary image by setting pixels with a value greater than `fThreshold` to white (255) and others to black (0). `fThreshold` is a parameter of the corresponding functions. To find a good value, you will most probably approach it via different attempts.

**Dynamic threshold value**

A dynamic threshold according to Otsu works perfectly for bimodal images. The histogram of a bimodal image has two main peaks and a good threshold value is most likely exactly in the middle. If you have a noisy image, you can use a Gaussian filter to shift it towards bimodal.

**Adaptive threshold value**
An adaptive threshold value uses variable threshold values for the whole image. See F_VN_AdaptiveThresholdExp [1163].

**Parameter**

**Input image**
The input image `ipSrcImage` may have one or three channels.

**Result image**
The result image `ipDestImage` returns the segmented binary image.

**Threshold value**
The threshold `fThreshold` defines the segmentation condition. It must be adjusted depending on the properties of the input image.

**Target value**
The target value `fMaxValue` defines which value should be given to the pixels that fulfil the threshold criterion. If the result image is to consist only of black and white pixels, then the maximum value of the element type of the input image must be set here. This corresponds to \((2^d) - 1\), where \(d\) represents the bit depth of the element type; e.g. \((2^8) - 1 = 255\) with 8-bit images.

**Threshold type**
The threshold type `eThresholdType` defines the algorithm according to which the threshold `fThreshold` is applied to the input image. All values of the enum EtcVnThresholdType [201] are supported.

**Application**
The binary segmentation of an 8-bit grayscale image in the center of the value range looks like this, for example:

```c
hr := F_VN_Threshold(
    ipSrcImage := ipImageIn,
    ipDestImage := ipImageRes,
    fThreshold := 128, // half value range: \((2^8)/2 = 128\)
    fMaxValue := 255, // max value of 8-bit image: \((2^8) - 1 = 255\)
    eThresholdType := TCVN_TT_BINARY,
    hrPrev := hr
);
```

**Related functions**
- F_VN_Threshold [1166] for binary segmentation on the basis of a threshold value
- F_VN_AdaptiveThreshold [1162] for binary, adaptive segmentation
- F_VN_CheckColorRange [1165] for binary segmentation on the basis of a color spectrum
- F_VN_ReferenceColorSimilarity [1097] for binary segmentation on the basis of a color model

**Required License**
TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

Version: 1.3
6.4.16.5 F_VN_WatershedSegmentationExp

Apply a marker-based watershed segmentation. (expert function)

Syntax

Definition:

```c
FUNCTION F_VN_WatershedSegmentationExp : HRESULT
VAR_INPUT
  ipSrcImage : ITCvNImage;
  ipDestImage : Reference To ITCvNImage;
  ipMarkers : ITCvNImage;
  hrPrev : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvNImage [383]</td>
<td>Source image (USINT, 3 channels)</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITCvNImage [383]</td>
<td>Destination image (DINT, 1 channel. An appropriate image will be created</td>
</tr>
<tr>
<td>ipMarkers</td>
<td>ITCvNImage [383]</td>
<td>Marker image (DINT, 1 channel)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.17 Keypoint Features

This group contains functions for finding and matching distinctive features in an image.

NOTE

Long computing duration

Many functions for finding keypoint features require a long computing duration. Therefore, pay attention to the set cycle time and if necessary use a watchdog [137] to secure the real-time behavior.
Functions

Key Points

- F_VN_KeyPointsAGAST(Exp) [1188]
- F_VN_KeyPointsFAST(Exp) [1197]
- F_VN_KeyPointsGFTT(Exp) [1199]
- F_VN_KeyPointsMSER(Exp) [1201]
- F_VN_KeyPointsSB(Exp) [1203]

Key points and descriptors

- F_VN_KeyPointsAndDescriptorsAKAZE(Exp) [1190]
- F_VN_KeyPointsAndDescriptorsBRISK(Exp) [1191]
- F_VN_KeyPointsAndDescriptorsaKAZE(Exp) [1193]
- F_VN_KeyPointsAndDescriptorsORB(Exp) [1195]

Descriptor matching

- F_VN_FilterGoodMatches [1170]
- F_VN_GetMatchCoordinates [1187]
- F_VN_MatchDescriptorsBF(Exp) [1205]
- F_VN_MatchDescriptorsFlannLsh(Exp) [1207]
- F_VN_MatchDescriptorsKnnBF(Exp) [1209]
- F_VN_MatchDescriptorsKnnFlannLsh(Exp) [1211]

Key Points Referenz Matching

- F_VN_FindReferenceKeyPointsInImage(Exp) [1171]
- F_VN_FindReferenceKeyPointsInImageAKAZE(Exp) [1173]
- F_VN_FindReferenceKeyPointsInImageBRISK(Exp) [1177]
- F_VN_FindReferenceKeyPointsInImageORB(Exp) [1183]

Miscellaneous

- F_VN_RegionsMSER(Exp) [1213]

6.4.17.1 F_VN_FilterGoodMatches

Filter the descriptor matches and return only good ones.

Syntax

Definition:

FUNCTION F_VN_FilterGoodMatches : HRESULT
VAR_INPUT
ipMatches : ITCvnContainer;
ipGoodMatches : Reference To ITCvnContainer;
fMaxDist : REAL;
fMaxKnnRatio : REAL;
hrPrev : HRESULT
TF7000 - TF7300
Version: 1.3

6.4.17.2  F_VN_FindReferenceKeyPointsInImage

Searches a reference image, represented by its keypoints and corresponding descriptors, in a given source image.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

FUNCTION F_VN_FindReferenceKeyPointsInImage : HRESULT
VAR_INPUT
   ipSrcKeyPoints  : ITcVnContainer;
   ipSrcDescriptors : ITcVnImage;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipMatches</td>
<td>ITcVnContainer [345]</td>
<td>Container with descriptor matches (ContainerType_Vector_TcVnMatch or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ContainerType_Vector_Vector_TcVnMatch)</td>
</tr>
<tr>
<td>ipGoodMatches</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container with good matches (ContainerType_Vector_TcVnMatch)</td>
</tr>
<tr>
<td>fMaxDist</td>
<td>REAL</td>
<td>Maximum allowed descriptor distance (-1 disables this filter criterion)</td>
</tr>
<tr>
<td>fMaxKnnRatio</td>
<td>REAL</td>
<td>Maximum allowed distance ratio between first and second best match ([0..1], used for knn match results only, -1 disables this filter criterion)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Matching

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

Intel 4-core Atom CPU
PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU

Tc3_Vision
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcKeyPoints</td>
<td>ITcVnContainer [345]</td>
<td>KeyPoints of the source image, e.g. (ContainerType_Vector_TcVnKeyPoint)</td>
</tr>
<tr>
<td>ipSrcDescriptors</td>
<td>ITcVnImage [383]</td>
<td>Descriptors of the source image KeyPoints</td>
</tr>
<tr>
<td>ipRefKeyPoints</td>
<td>ITcVnContainer [345]</td>
<td>KeyPoints of the reference image, e.g. (ContainerType_Vector_TcVnKeyPoint)</td>
</tr>
<tr>
<td>ipRefDescriptors</td>
<td>ITcVnImage [383]</td>
<td>Descriptors of the reference image KeyPoints</td>
</tr>
<tr>
<td>nRefImageWidth</td>
<td>UDINT</td>
<td>Width of the reference image in pixels</td>
</tr>
<tr>
<td>nRefImageHeight</td>
<td>UDINT</td>
<td>Height of the reference image in pixels</td>
</tr>
<tr>
<td>ipEdgePoints</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container with the 4 edge points of the reference image transformed into the coordinates of the source image (ContainerType_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>fMaxDist</td>
<td>REAL</td>
<td>Maximum allowed descriptor distance (-1 disables this filter criterion)</td>
</tr>
<tr>
<td>fMaxKnnRatio</td>
<td>REAL</td>
<td>Maximum allowed distance ratio between first and second best match ([0..1], -1 disables this filter criterion)</td>
</tr>
<tr>
<td>eNormType</td>
<td>ETcVnNormalizationType [195]</td>
<td>Normalization type used for descriptor matching (HAMMING recommended for AKAZE, ORB, and BRISK, HAMMING2 is recommended for ORB if the ORB nBriefPoints parameter is 3 or 4.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aPerspectiveTransform</td>
<td>TcVnMatrix3x3_LREAL [153]</td>
<td>Returns the perspective transformation matrix, which transforms the reference points to the source points</td>
</tr>
<tr>
<td>nNumberOfGoodMatches</td>
<td>UDINT</td>
<td>Return the number of good matches</td>
</tr>
<tr>
<td>nNumberOfInliers</td>
<td>UDINT</td>
<td>Return the number of inlier matches</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]
### Required License
TC3 Vision Matching

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

#### 6.4.17.3 F_VN_FindReferenceKeyPointsInImageAKAZE

Searches a reference image, represented by its AKAZE keypoints and corresponding descriptors, in a given source image (uses the default TcVnParamsAKAZE parameters). Can use available TwinCAT Job Tasks for executing parallel code regions.

**Syntax**

**Definition:**

```
FUNCTION F_VN_FindReferenceKeyPointsInImageAKAZE : HRESULT
VAR_INPUT
    ipSrcImage : ITCvImage;
    ipRefKeyPoints : ITCvContainer;
    ipRefDescriptors : ITCvImage;
    nRefImageWidth : UDINT;
    nRefImageHeight : UDINT;
    ipEdgePoints : Reference To ITCvContainer;
    fMaxDist : REAL;
    fMaxKnnRatio : REAL;
    hrPrev : HRESULT;
END_VAR
VAR_OUTPUT
    aPerspectiveTransform : TcVnMatrix3x3_LREAL;
    nNumberOfGoodMatches : UDINT;
    nNumberOfInliers : UDINT;
END_VAR
```

```c
F_VN_FindReferenceKeyPointsInImageAKAZE
```
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCnImage ⟨383⟩</td>
<td>Source image</td>
</tr>
<tr>
<td>ipRefKeyPoints</td>
<td>ITCnContainer ⟨345⟩</td>
<td>KeyPoints of the reference image, e.g. computed with F_VN_KEYPointsAndDescriptorsAKAZE (ContainerType_Vector_TcVnKeyPoint)</td>
</tr>
<tr>
<td>ipRefDescriptor</td>
<td>ITCnImage ⟨383⟩</td>
<td>Descriptors of the reference image KeyPoints, e.g. computed with F_VN_KEYPointsAndDescriptorsAKAZE</td>
</tr>
<tr>
<td>nRefImageWidth</td>
<td>UDINT</td>
<td>Width of the reference image in pixels</td>
</tr>
<tr>
<td>nRefImageHeight</td>
<td>UDINT</td>
<td>Height of the reference image in pixels</td>
</tr>
<tr>
<td>ipEdgePoints</td>
<td>Reference To ITCnContainer ⟨345⟩</td>
<td>Returns a container with the 4 edge points of the reference image transformed into ipSrcImage coordinates (ContainerType_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>fMaxDist</td>
<td>REAL</td>
<td>Maximum allowed descriptor distance (-1 disables this filter criterion)</td>
</tr>
<tr>
<td>fMaxKnnRatio</td>
<td>REAL</td>
<td>Maximum allowed distance ratio between first and second best match ([0..1], -1 disables this filter criterion)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT ⟨135⟩</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aPerspectiveTranform</td>
<td>TcVnMatrix3x3_LREAL ⟨153⟩</td>
<td>Returns the perspective transformation matrix, which transforms the reference points to the source points</td>
</tr>
<tr>
<td>nNumberOfGoodMatches</td>
<td>UDINT</td>
<td>Return the number of good matches</td>
</tr>
<tr>
<td>nNumberOfInliers</td>
<td>UDINT</td>
<td>Return the number of inlier matches</td>
</tr>
</tbody>
</table>

### Return value

HRESULT ⟨135⟩

### Required License

TC3 Vision Matching

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.17.4  F_VN_FindReferenceKeyPointsInImageAKAZEEExp

Searches a reference image, represented by its AKAZE keypoints and corresponding descriptors, in a given source image. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

FUNCTION F_VN_FindReferenceKeyPointsInImageAKAZEEExp : HRESULT
VAR_INPUT
   ipSrcImage : ITcVnImage;
   ipRefKeyPoints : ITcVnContainer;
   ipRefDescriptors : ITcVnImage;
   nRefImageHeight : UDINT;
   ipEdgePoints : Reference To ITcVnContainer;
   fMaxDist : REAL;
   fMaxKnnRatio : REAL;
   eNormType : ETcVnNormalizationType;
   ipSrcImageMask : ITcVnImage;
   stAkazeParams : Reference To TcVnParamsAKAZE;
   eAlgorithm : ETcVnEstimationAlgorithm;
   fReprojThreshold : LREAL;
   nMaxIterations : UDINT;
   fConfidence : LREAL;
   hrPrev : HRESULT;
END_VAR
VAR_OUTPUT
   aPerspectiveTransform : TcVnMatrix3x3_LREAL;
   nNumberOfGoodMatches : UDINT;
   nNumberOfInliers : UDINT;
END_VAR
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipRefKeyPoints</td>
<td>ITcVnContainer [345]</td>
<td>KeyPoints of the reference image, e.g. computed with F_VN_KeyPointsAndDescriptorsAKAZE (ContainerType_Vector_TcVnKeyPoint)</td>
</tr>
<tr>
<td>ipRefDescriptors</td>
<td>ITcVnImage [383]</td>
<td>Descriptors of the reference image KeyPoints, e.g. computed with F_VN_KeyPointsAndDescriptorsAKAZE</td>
</tr>
<tr>
<td>nRefImageWidth</td>
<td>UDINT</td>
<td>Width of the reference image in pixels</td>
</tr>
<tr>
<td>nRefImageHeight</td>
<td>UDINT</td>
<td>Height of the reference image in pixels</td>
</tr>
<tr>
<td>ipEdgePoints</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container with the 4 edge points of the reference image transformed into ipSrcImage coordinates (ContainerType_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>fMaxDist</td>
<td>REAL</td>
<td>Maximum allowed descriptor distance (-1 disables this filter criterion)</td>
</tr>
<tr>
<td>fMaxKnnRatio</td>
<td>REAL</td>
<td>Maximum allowed distance ratio between first and second best match ([0..1], -1 disables this filter criterion)</td>
</tr>
<tr>
<td>eNormType</td>
<td>ETcVnNormalizationType [195]</td>
<td>Normalization type used for descriptor matching</td>
</tr>
<tr>
<td>ipSrcImageMask</td>
<td>ITcVnImage [383]</td>
<td>Mask to specify, where to look for keypoints in ipSrcImage (set to 0 if no mask required)</td>
</tr>
<tr>
<td>stAkazeParams</td>
<td>Reference To TcVnParamsAKAZE [211]</td>
<td>Parameters to configure the keypoint and descriptor computation for ipSrcImage (resulting descriptors must be compatible to ipRefDescriptors!)</td>
</tr>
<tr>
<td>eAlgorithm</td>
<td>ETcVnEstimationAlgorithm [187]</td>
<td>Estimation algorithm used for computing the perspective transformation between the two point sets</td>
</tr>
<tr>
<td>fReprojThreshold</td>
<td>LREAL</td>
<td>Maximum allowed reprojection error to treat a point pair as an inlier (only used if eAlgorithm is RANSAC, RHO)</td>
</tr>
<tr>
<td>nMaxIterations</td>
<td>UDINT</td>
<td>Maximum number of RANSAC iterations</td>
</tr>
<tr>
<td>fConfidence</td>
<td>LREAL</td>
<td>Confidence (0..1)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aPerspectiveTransform</td>
<td>TcVnMatrix3x3_LREAL [153]</td>
<td>Returns the perspective transformation matrix, which transforms the reference points to the source points</td>
</tr>
<tr>
<td>nNumberOfGoodMatches</td>
<td>UDINT</td>
<td>Return the number of good matches</td>
</tr>
<tr>
<td>nNumberOfInliers</td>
<td>UDINT</td>
<td>Return the number of inlier matches</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Matching
## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.17.5  
**F_VN_FindReferenceKeyPointsInImageBRISK**

Searches a reference image, represented by its BRISK keypoints and corresponding descriptors, in a given source image (uses the default TcVnParamsBRISK parameters). Can use available TwinCAT Job Tasks for executing parallel code regions.

#### Syntax

**Definition:**

```c
FUNCTION F_VN_FindReferenceKeyPointsInImageBRISK : HRESULT
VAR_INPUT
    ipSrcImage : ITCvnImage;
    ipRefKeyPoints : ITCvnContainer;
    ipRefDescriptors : ITCvnImage;
    nRefImageWidth : UDINT;
    nRefImageHeight : UDINT;
    ipEdgePoints : Reference To ITCvnContainer;
    fMaxDist : REAL;
    fMaxKnnRatio : REAL;
    hrPrev : HRESULT;
END_VAR
VAR_OUTPUT
    aPerspectiveTransform : TcVnMatrix3x3_LREAL;
    nNumberOfGoodMatches : UDINT;
    nNumberOfInliers : UDINT;
END_VAR
```

Searches a reference image, represented by its BRISK keypoints and corresponding descriptors, in a given source image (uses the default TcVnParamsBRISK parameters). Can use available TwinCAT Job Tasks for executing parallel code regions.

#### Syntax

**Definition:**

```c
FUNCTION F_VN_FindReferenceKeyPointsInImageBRISK : HRESULT
VAR_INPUT
    ipSrcImage : ITCvnImage;
    ipRefKeyPoints : ITCvnContainer;
    ipRefDescriptors : ITCvnImage;
    nRefImageWidth : UDINT;
    nRefImageHeight : UDINT;
    ipEdgePoints : Reference To ITCvnContainer;
    fMaxDist : REAL;
    fMaxKnnRatio : REAL;
    hrPrev : HRESULT;
END_VAR
VAR_OUTPUT
    aPerspectiveTransform : TcVnMatrix3x3_LREAL;
    nNumberOfGoodMatches : UDINT;
    nNumberOfInliers : UDINT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [1383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipRefKeyPoints</td>
<td>ITcVnContainer [345]</td>
<td>KeyPoints of the reference image, e.g. computed with F_VN_KeyPointsAndDescriptorsBRISK (ContainerType_Vector_TcVnKeyPoint)</td>
</tr>
<tr>
<td>ipRefDescriptors</td>
<td>ITcVnImage [1383]</td>
<td>Descriptors of the reference image KeyPoints, e.g. computed with F_VN_KeyPointsAndDescriptorsBRISK</td>
</tr>
<tr>
<td>nRefImageWidth</td>
<td>UDINT</td>
<td>Width of the reference image in pixels</td>
</tr>
<tr>
<td>nRefImageHeight</td>
<td>UDINT</td>
<td>Height of the reference image in pixels</td>
</tr>
<tr>
<td>ipEdgePoints</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container with the 4 edge points of the reference image transformed into ipSrcImage coordinates (ContainerType_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>fMaxDist</td>
<td>REAL</td>
<td>Maximum allowed descriptor distance (-1 disables this filter criterion)</td>
</tr>
<tr>
<td>fMaxKnnRatio</td>
<td>REAL</td>
<td>Maximum allowed distance ratio between first and second best match ([0..1], -1 disables this filter criterion)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aPerspectiveTransform</td>
<td>TcVnMatrix3x3_LREAL [153]</td>
<td>Returns the perspective transformation matrix, which transforms the reference points to the source points</td>
</tr>
<tr>
<td>nNumberOfGoodMatches</td>
<td>UDINT</td>
<td>Return the number of good matches</td>
</tr>
<tr>
<td>nNumberOfInliers</td>
<td>UDINT</td>
<td>Return the number of inlier matches</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Matching

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.17.6  F_VN_FindReferenceKeyPointsInImageBRISKExp

Searches a reference image, represented by its BRISK keypoints and corresponding descriptors, in a given source image. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_FindReferenceKeyPointsInImageBRISKExp : HRESULT
VAR_INPUT
    ipSrcImage : ITCvImage;
    ipRefKeyPoints : ITCvContainer;
    ipRefDescriptors : ITCvImage;
    nRefImageWidth : UDINT;
    nRefImageHeight : UDINT;
    ipEdgePoints : Reference To ITCvContainer;
    fMaxDist : REAL;
    fMaxKnnRatio : REAL;
    eNormType : ETCvNormalizationType;
    ipSrcImageMask : ITCvImage;
    stBriskParams : Reference To TcVnParamsBRISK;
    eAlgorithm : ETCvEstimationAlgorithm;
    fReprojThreshold : LREAL;
    nMaxIterations : UDINT;
    fConfidence : LREAL;
    hrPrev : HRESULT;
END_VAR
VAR_OUTPUT
    aPerspectiveTransform : TcVnMatrix3x3_LREAL;
    nNumberOfGoodMatches : UDINT;
    nNumberOfInliers : UDINT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage[383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipRefKeyPoints</td>
<td>ITcVnContainer[345]</td>
<td>KeyPoints of the reference image, e.g. computed with F_VN_KeyPointsAndDescriptorsBRISK (ContainerType_Vector_TcVnKeyPoint)</td>
</tr>
<tr>
<td>ipRefDescriptors</td>
<td>ITcVnImage[383]</td>
<td>Descriptors of the reference image KeyPoints, e.g. computed with F_VN_KeyPointsAndDescriptorsBRISK (ContainerType_Vector_TcVnKeyPoint2_REAL)</td>
</tr>
<tr>
<td>nRefImageWidth</td>
<td>UDINT</td>
<td>Width of the reference image in pixels</td>
</tr>
<tr>
<td>nRefImageHeight</td>
<td>UDINT</td>
<td>Height of the reference image in pixels</td>
</tr>
<tr>
<td>ipEdgePoints</td>
<td>Reference To ITcVnContainer[345]</td>
<td>Returns a container with the 4 edge points of the reference image transformed into ipSrcImage coordinates (ContainerType_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>fMaxDist</td>
<td>REAL</td>
<td>Maximum allowed descriptor distance (-1 disables this filter criterion)</td>
</tr>
<tr>
<td>fMaxKnnRatio</td>
<td>REAL</td>
<td>Maximum allowed distance ratio between first and second best match ([0..1], -1 disables this filter criterion)</td>
</tr>
<tr>
<td>eNormType</td>
<td>ETcVnNormalizationType[195]</td>
<td>Normalization type used for descriptor matching</td>
</tr>
<tr>
<td>ipSrcImageMask</td>
<td>ITcVnImage[383]</td>
<td>Mask to specify, where to look for keypoints in ipSrcImage (set to 0 if no mask required)</td>
</tr>
<tr>
<td>stBriskParams</td>
<td>Reference To TcVnParamsBRISK[215]</td>
<td>Parameters to configure the keypoint and descriptor computation for ipSrcImage (resulting descriptors must be compatible to ipRefDescriptors!)</td>
</tr>
<tr>
<td>eAlgorithm</td>
<td>ETcVnEstimationAlgorithm[187]</td>
<td>Estimation algorithm used for computing the perspective transformation between the two point sets</td>
</tr>
<tr>
<td>fReprojThreshold</td>
<td>LREAL</td>
<td>Maximum allowed reprojection error to treat a point pair as an inlier (only used if eAlgorithm is RANSAC, RHO)</td>
</tr>
<tr>
<td>nMaxIterations</td>
<td>UDINT</td>
<td>Maximum number of RANSAC iterations</td>
</tr>
<tr>
<td>fConfidence</td>
<td>LREAL</td>
<td>Confidence (0..1)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aPerspectiveTransform</td>
<td>TcVnMatrix3x3_LREAL[153]</td>
<td>Returns the perspective transformation matrix, which transforms the reference points to the source points</td>
</tr>
<tr>
<td>nNumberOfGoodMatches</td>
<td>UDINT</td>
<td>Return the number of good matches</td>
</tr>
<tr>
<td>nNumberOfInliers</td>
<td>UDINT</td>
<td>Return the number of inlier matches</td>
</tr>
</tbody>
</table>

## Return value

HRESULT[135]

## Required License

TC3 Vision Matching
### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

#### 6.4.17.7 F_VN_FindReferenceKeyPointsInImageExp

**F_VN_FindReferenceKeyPointsInImageExp**

```plaintext
FUNCTION F_VN_FindReferenceKeyPointsInImageExp : HRESULT
VAR_INPUT
    ipSrcKeyPoints : ITCvNContainer;
    ipSrcDescriptors : ITCvNImage;
    ipRefKeyPoints : ITCvNContainer;
    ipRefDescriptors : ITCvNImage;
    nRefImageWidth : UDINT;
    nRefImageHeight : UDINT;
    ipEdgePoints : Reference To ITCvNContainer;
    fMaxDist : REAL;
    fMaxKnnRatio : REAL;
    eNormType : ETCvNNormalizationType;
    eAlgorithm : ETCvNEstimationAlgorithm;
    fReprojThreshold : REAL;
    nMaxIterations : UDINT;
    fConfidence : LREAL;
    ipGoodMatches : Reference To ITCvNContainer;
    ipInlierMask : Reference To ITCvNContainer;
    hrPrev : HRESULT;
END_VAR
VAR_OUTPUT
    aPerspectiveTransform : TcVnMatrix3x3_LREAL;
    nNumberOfGoodMatches : UDINT;
    nNumberOfInliers : UDINT;
END_VAR
```

Searches a reference image, represented by its keypoints and corresponding descriptors, in a given source image. (expert function)

Can use available TwinCAT Job Tasks for executing parallel code regions.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_FindReferenceKeyPointsInImageExp : HRESULT
VAR_INPUT
    ipSrcKeyPoints : ITCvNContainer;
    ipSrcDescriptors : ITCvNImage;
    ipRefKeyPoints : ITCvNContainer;
    ipRefDescriptors : ITCvNImage;
    nRefImageWidth : UDINT;
    nRefImageHeight : UDINT;
    ipEdgePoints : Reference To ITCvNContainer;
    fMaxDist : REAL;
    fMaxKnnRatio : REAL;
    eNormType : ETCvNNormalizationType;
    eAlgorithm : ETCvNEstimationAlgorithm;
    fReprojThreshold : REAL;
    nMaxIterations : UDINT;
    fConfidence : LREAL;
    ipGoodMatches : Reference To ITCvNContainer;
    ipInlierMask : Reference To ITCvNContainer;
    hrPrev : HRESULT;
END_VAR
VAR_OUTPUT
    aPerspectiveTransform : TcVnMatrix3x3_LREAL;
    nNumberOfGoodMatches : UDINT;
    nNumberOfInliers : UDINT;
END_VAR
```

**System Requirements**

- **Development environment:** TwinCAT V3.1.4024.17 or later
- **Target platform:** PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU
- **PLC libraries to include:** Tc3_Vision
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcKeyPoints</td>
<td>ITcVnContainer[1] 345</td>
<td>KeyPoints of the source image, e.g. (ContainerType_Vector_TcVnKeyPoint)</td>
</tr>
<tr>
<td>ipSrcDescriptor</td>
<td>ITcVnImage[1] 383</td>
<td>Descriptors of the source image KeyPoints</td>
</tr>
<tr>
<td>ipRefKeyPoints</td>
<td>ITcVnContainer[1] 345</td>
<td>KeyPoints of the reference image (ContainerType_Vector_TcVnKeyPoint)</td>
</tr>
<tr>
<td>ipRefDescriptors</td>
<td>ITcVnImage[1] 383</td>
<td>Descriptors of the reference image KeyPoints</td>
</tr>
<tr>
<td>nRefImageWidth</td>
<td>UDINT</td>
<td>Width of the reference image in pixels</td>
</tr>
<tr>
<td>nRefImageHeight</td>
<td>UDINT</td>
<td>Height of the reference image in pixels</td>
</tr>
<tr>
<td>ipEdgePoints</td>
<td>Reference To ITcVnContainer[1] 345</td>
<td>Returns a container with the 4 edge points of the reference image transformed into the coordinates of the source image (ContainerType_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>fMaxDist</td>
<td>REAL</td>
<td>Maximum allowed descriptor distance (-1 disables this filter criterion)</td>
</tr>
<tr>
<td>fMaxKnnRatio</td>
<td>REAL</td>
<td>Maximum allowed distance ratio between first and second best match ([0..1], -1 disables this filter criterion)</td>
</tr>
<tr>
<td>eNormType</td>
<td>ETcVnNormalizationType[1] 195</td>
<td>Normalization type used for descriptor matching (HAMMING recommended for AKAZE, ORB, and BRISK. HAMMING2 is recommended for ORB if the ORB nBriefPoints parameter is 3 or 4.)</td>
</tr>
<tr>
<td>eAlgorithm</td>
<td>ETcVnEstimationAlgorithm[1] 187</td>
<td>Estimation algorithm used for computing the perspective transformation between the two point sets</td>
</tr>
<tr>
<td>fReprojThreshold</td>
<td>LREAL</td>
<td>Maximum allowed reprojection error to treat a point pair as an inlier (only used if eAlgorithm is RANSAC, RHO)</td>
</tr>
<tr>
<td>nMaxIterations</td>
<td>UDINT</td>
<td>Maximum number of RANSAC iterations</td>
</tr>
<tr>
<td>fConfidence</td>
<td>LREAL</td>
<td>Confidence (0..1)</td>
</tr>
<tr>
<td>ipGoodMatches</td>
<td>Reference To ITcVnContainer[1] 345</td>
<td>Returns a container with good matches (optional, set to 0 if not required; ContainerType_Vector_TcVnDMatch)</td>
</tr>
<tr>
<td>ipInlierMask</td>
<td>Reference To ITcVnContainer[1] 345</td>
<td>Returns a mask marking the inliers (optional, set to 0 if not required; ContainerType_Vector_SINT; only for RANSAC, LMEDS)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[1] 135</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aPerspectiveTransform</td>
<td>TcVnMatrix3x3_LREAL[1] 153</td>
<td>Returns the perspective transformation matrix, which transforms the reference points to the source points</td>
</tr>
<tr>
<td>nNumberOfGoodMatches</td>
<td>UDINT</td>
<td>Return the number of good matches</td>
</tr>
<tr>
<td>nNumberOfInliers</td>
<td>UDINT</td>
<td>Return the number of inlier matches</td>
</tr>
</tbody>
</table>

### Return value

HRESULT[1] 135
Required License

TC3 Vision Matching

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.17.8 **F_VN_FindReferenceKeyPointsInImageORB**

Searches a reference image, represented by its ORB keypoints and corresponding descriptors, in a given source image (uses the default TcVnParamsORB parameters).

Can use available TwinCAT Job Tasks for executing parallel code regions.

**Syntax**

**Definition:**

```
FUNCTION F_VN_FindReferenceKeyPointsInImageORB : HRESULT
VAR_INPUT
    ipSrcImage : ITCvImage;
    ipRefKeyPoints : ITCvContainer;
    ipRefDescriptors : ITCvImage;
    nRefImageWidth : UDINT;
    nRefImageHeight : UDINT;
    ipEdgePoints : Reference To ITCvContainer;
    fMaxDist : REAL;
    fMaxKnnRatio : REAL;
    hrPrev : HRESULT;
END_VAR
VAR_OUTPUT
    aPerspectiveTransform : TcVnMatrix3x3_LREAL;
    nNumberOfGoodMatches : UDINT;
    nNumberOfInliers : UDINT;
END_VAR
```
**API reference**

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage[383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipRefKeyPoints</td>
<td>ITcVnContainer[345]</td>
<td>KeyPoints of the reference image, e.g. computed with F_VN_KeyPointsAndDescriptorsORB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(ContainerType_Vector_TcVnKeyPoint)</td>
</tr>
<tr>
<td>ipRefDescriptor</td>
<td>ITcVnImage[383]</td>
<td>Descriptors of the reference image KeyPoints, e.g. computed with F_VN_KeyPointsAndDescriptorsORB</td>
</tr>
<tr>
<td>nRefImageWidth</td>
<td>UDINT</td>
<td>Width of the reference image in pixels</td>
</tr>
<tr>
<td>nRefImageHeight</td>
<td>UDINT</td>
<td>Height of the reference image in pixels</td>
</tr>
<tr>
<td>ipEdgePoints</td>
<td>Reference To ITcVnContainer[345]</td>
<td>Returns a container with the 4 edge points of the reference image transformed into ipSrcImage coordinates (ContainerType_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>fMaxDist</td>
<td>REAL</td>
<td>Maximum allowed descriptor distance (-1 disables this filter criterion)</td>
</tr>
<tr>
<td>fMaxKnnRatio</td>
<td>REAL</td>
<td>Maximum allowed distance ratio between first and second best match ([0..1], -1 disables this filter criterion)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aPerspectiveTr</td>
<td>TcVnMatrix3x3_LREAL[153]</td>
<td>Returns the perspective transformation matrix, which transforms the reference points to the source points</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nNumberOfGoodMatches</td>
<td>UDINT</td>
<td>Return the number of good matches</td>
</tr>
<tr>
<td>nNumberOfInliers</td>
<td>UDINT</td>
<td>Return the number of inlier matches</td>
</tr>
</tbody>
</table>

### Return value

HRESULT[135]

### Required License

TC3 Vision Matching

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.17.9  F_VN_FindReferenceKeyPointsInImageORBExp

F_VN_FindReferenceKeyPointsInImageORBExp

Definition:

Searches a reference image, represented by its ORB keypoints and corresponding descriptors, in a given source image. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

FUNCTION F_VN_FindReferenceKeyPointsInImageORBExp : HRESULT
VAR_INPUT
    ipSrcImage : ITCvNImage;
    ipRefKeyPoints : ITCvNContainer;
    ipRefDescriptors : ITCvNImage;
    nRefImageWidth : UDINT;
    nRefImageHeight : UDINT;
    ipEdgePoints : Reference To ITCvNContainer;
    fMaxDist : REAL;
    fMaxKnnRatio : REAL;
    eNormType : ETcVnNormalizationType;
    ipSrcImageMask : ITCvNImage;
    stOrbParams : Reference To TcVnParamsORB;
    eAlgorithm : ETcVnEstimationAlgorithm;
    fReprojThreshold : LREAL;
    fConfidence : LREAL;
    hrPrev : HRESULT;
END_VAR

VAR_OUTPUT
    aPerspectiveTransform : TcVnMatrix3x3_LREAL;
    nNumberOfInliers : UDINT;
    nNumberOfGoodMatches : UDINT;
END_VAR
# Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage <img src="#" alt="383" /></td>
<td>Source image</td>
</tr>
<tr>
<td>ipRefKeyPoints</td>
<td>ITcVnContainer <img src="#" alt="345" /></td>
<td>KeyPoints of the reference image, e.g. computed with F_VN_KeyPointsAndDescriptorsORB (ContainerType_Vector_TcVnKeyPoint)</td>
</tr>
<tr>
<td>ipRefDescriptors</td>
<td>ITcVnImage <img src="#" alt="383" /></td>
<td>Descriptors of the reference image KeyPoints, e.g. computed with F_VN_KeyPointsAndDescriptorsORB</td>
</tr>
<tr>
<td>nRefImageWidth</td>
<td>UDINT</td>
<td>Width of the reference image in pixels</td>
</tr>
<tr>
<td>nRefImageHeight</td>
<td>UDINT</td>
<td>Height of the reference image in pixels</td>
</tr>
<tr>
<td>ipEdgePoints</td>
<td>Reference To ITcVnContainer <img src="#" alt="345" /></td>
<td>Returns a container with the 4 edge points of the reference image transformed into ipSrcImage coordinates (ContainerType_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>fMaxDist</td>
<td>REAL</td>
<td>Maximum allowed descriptor distance (-1 disables this filter criterion)</td>
</tr>
<tr>
<td>fMaxKnnRatio</td>
<td>REAL</td>
<td>Maximum allowed distance ratio between first and second best match ([0..1], -1 disables this filter criterion)</td>
</tr>
<tr>
<td>eNormType</td>
<td>ETcVnNormalizationType <img src="#" alt="195" /></td>
<td>Normalization type used for descriptor matching (HAMMING recommended for ORB, HAMMING2 if the ORB nBriefPoints parameter is 3 or 4)</td>
</tr>
<tr>
<td>ipSrcImageMask</td>
<td>ITcVnImage <img src="#" alt="383" /></td>
<td>Mask to specify, where to look for keypoints in ipSrcImage (set to 0 if no mask required)</td>
</tr>
<tr>
<td>stOrbParams</td>
<td>Reference To TcVnParamsORB <img src="#" alt="218" /></td>
<td>Parameters to configure the keypoint and descriptor computation for ipSrcImage (resulting descriptors must be compatible to ipRefDescriptors!)</td>
</tr>
<tr>
<td>eAlgorithm</td>
<td>ETcVnEstimationAlgorithm <img src="#" alt="187" /></td>
<td>Estimation algorithm used for computing the perspective transformation between the two point sets</td>
</tr>
<tr>
<td>fReprojThreshold</td>
<td>LREAL</td>
<td>Maximum allowed reprojection error to treat a point pair as an inlier (only used if eAlgorithm is RANSAC, RHO)</td>
</tr>
<tr>
<td>nMaxIterations</td>
<td>UDINT</td>
<td>Maximum number of RANSAC iterations</td>
</tr>
<tr>
<td>fConfidence</td>
<td>LREAL</td>
<td>Confidence (0..1)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT <img src="#" alt="135" /></td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

# Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aPerspectiveTranform</td>
<td>TcVnMatrix3x3_LREAL <img src="#" alt="153" /></td>
<td>Returns the perspective transformation matrix, which transforms the reference points to the source points</td>
</tr>
<tr>
<td>nNumberOfGoodMatches</td>
<td>UDINT</td>
<td>Return the number of good matches</td>
</tr>
<tr>
<td>nNumberOfInliers</td>
<td>UDINT</td>
<td>Return the number of inlier matches</td>
</tr>
</tbody>
</table>

# Return value

HRESULT ![135](#)

# Required License

TC3 Vision Matching
6.4.17.10 F_VN_GetMatchCoordinates

Return the coordinates of keypoints that match each other.

Syntax

Definition:

FUNCTION F_VN_GetMatchCoordinates : HRESULT
VAR_INPUT
  ipQueryKeyPoints : ITcVnContainer;
  ipTrainKeyPoints : ITcVnContainer;
  ipMatches : ITcVnContainer;
  ipQueryCoordinates : Reference To ITcVnContainer;
  ipTrainCoordinates : Reference To ITcVnContainer;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipQueryKeyPoints</td>
<td>ITcVnContainer [*345]</td>
<td>Container with query Keypoints (CTcVnContainer_Vector_TcVnKeyPoint)</td>
</tr>
<tr>
<td>ipTrainKeyPoints</td>
<td>ITcVnContainer [*345]</td>
<td>Container with train Keypoints (CTcVnContainer_Vector_TcVnKeyPoint)</td>
</tr>
<tr>
<td>ipMatches</td>
<td>ITcVnContainer [*345]</td>
<td>Container with matches between query and train Keypoints (CTcVnContainer_Vector_TcVnDMatch).</td>
</tr>
<tr>
<td>ipQueryCoordinates</td>
<td>Reference To ITcVnContainer [*345]</td>
<td>Returns a container with the coordinates of keypoints from the query Keypoints that exist in ipMatches (CTcVnContainer_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>ipTrainCoordinates</td>
<td>Reference To ITcVnContainer [*345]</td>
<td>Returns a container with the coordinates of keypoints from the training Keypoints that exist in ipMatches (CTcVnContainer_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [*135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [*135]

Required License

TC3 Vision Matching
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.17.11 F_VN_KeyPointsAGAST

Detected keypoints using the AGAST method. Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

FUNCTION F_VN_KeyPointsAGAST : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipKeyPoints : Reference To ITcVnContainer;
    hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvNImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipKeyPoints</td>
<td>Reference To ITCvNContainer [345]</td>
<td>Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Matching

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.17.12 F_VN_KeyPointsAGASTEpxp

F_VN_KeyPointsAGASTEpxp

ipSrcImage : ITcVnImage
ipKeyPoints : Reference To ITcVnContainer
ipMask : ITcVnImage
stParams : Reference To TcVnParamsAGAST
hrPrev : HRESULT

F_VN_KeyPointsAGASTEpxp

Detects keypoints using the AGAST method. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

FUNCTION F_VN_KeyPointsAGASTEpxp : HRESULT
VAR_INPUT
   ipSrcImage : ITcVnImage;
   ipKeyPoints : Reference To ITcVnContainer;
   ipMask : ITcVnImage;
   stParams : Reference To TcVnParamsAGAST;
   hrPrev : HRESULT;
END_VAR

_inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipKeyPoints</td>
<td>Reference To ITcVnContainer</td>
<td>Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITcVnImage [383]</td>
<td>Mask to specify, where to look for keypoints (set to 0 if no mask required)</td>
</tr>
<tr>
<td>stParams</td>
<td>Reference To TcVnParamsAGAST</td>
<td>Additional expert parameters</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

_Return value

HRESULT [135]

Required License

TC3 Vision Matching

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.17.13  F_VN_KeyPointsAndDescriptorsAKAZE

F_VN_KeyPointsAndDescriptorsAKAZE

<table>
<thead>
<tr>
<th>ipSrcImage</th>
<th>ITCvImage</th>
<th>HRESULT</th>
<th>F_VN_KeyPointsAndDescriptorsAKAZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipKeyPoints</td>
<td>Reference To ITCvContainer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ipDescriptors</td>
<td>Reference To ITCvImage</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Detects keypoints and compute descriptors using the AKAZE method. Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

FUNCTION F_VN_KeyPointsAndDescriptorsAKAZE : HRESULT

VAR_INPUT
   ipSrcImage : ITCvImage;
   ipKeyPoints : Reference To ITCvContainer;
   ipDescriptors : Reference To ITCvImage;
   hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipKeyPoints</td>
<td>Reference To ITCvContainer [345]</td>
<td>Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>ipDescriptors</td>
<td>Reference To ITCvImage [383]</td>
<td>Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Matching

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.17.14  F_VN_KeyPointsAndDescriptorsAKAZExp

F_VN_KeyPointsAndDescriptorsAKAZExp

<table>
<thead>
<tr>
<th>ipSrcImage</th>
<th>ITCvImage</th>
<th>HRESULT</th>
<th>F_VN_KeyPointsAndDescriptorsAKAZExp</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipKeyPoints</td>
<td>Reference To ITCvContainer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ipDescriptors</td>
<td>Reference To ITCvImage</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ipMask</td>
<td>ITCvImage</td>
<td></td>
<td></td>
</tr>
<tr>
<td>stParams</td>
<td>Reference To TcVnParamsAKAZE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1190  Version: 1.3  TF7000 - TF7300
Detects keypoints and compute descriptors using the AKAZE method. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

**Definition:**

```plaintext
FUNCTION F_VN_KeyPointsAndDescriptorsAKAZEExp : HRESULT
VAR_INPUT
  ipSrcImage    : ITcVnImage;
  ipKeyPoints   : Reference To ITcVnContainer;
  ipDescriptors : Reference To ITcVnImage;
  ipMask        : ITcVnImage;
  stParams      : Reference To TcVnParamsAKAZE;
  hrPrev        : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image</td>
</tr>
<tr>
<td>ipKeyPoints</td>
<td>Reference To ITcVnContainer</td>
<td>Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>ipDescriptors</td>
<td>Reference To ITcVnImage</td>
<td>Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITcVnImage</td>
<td>Mask to specify, where to look for keypoints (set to 0 if no mask required)</td>
</tr>
<tr>
<td>stParams</td>
<td>Reference To TcVnParamsAKAZE</td>
<td>Additional expert parameters</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Matching

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.17.15 F_VN_KeyPointsAndDescriptorsBRISK

Detects keypoints and compute descriptors using the BRISK method.
Can use available TwinCAT Job Tasks for executing parallel code regions.
### Syntax

#### Definition:

```c
FUNCTION F_VN_KeyPointsAndDescriptorsBRISK : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  ipKeyPoints : Reference To ITcVnContainer;
  ipDescriptors : Reference To ITcVnImage;
  hrPrev : HRESULT;
END_VAR
```

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image</td>
</tr>
<tr>
<td>ipKeyPoints</td>
<td>Reference To ITcVnContainer</td>
<td>Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>ipDescriptors</td>
<td>Reference To ITcVnImage</td>
<td>Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

#### Return value

```c
HRESULT [135]
```

### Required License

TC3 Vision Matching

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.17.16 F_VN_KeyPointsAndDescriptorsBRISKExp

Detects keypoints and compute descriptors using the BRISK method. (expert function)

Can use available TwinCAT Job Tasks for executing parallel code regions.

#### Syntax

#### Definition:

```c
FUNCTION F_VN_KeyPointsAndDescriptorsBRISKExp : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  ipKeyPoints : Reference To ITcVnContainer;
  ipDescriptors : Reference To ITcVnImage;
  ipMask : ITcVnImage;
  stParams : Reference To TcVnParamsBRISK;
  hrPrev : HRESULT;
END_VAR
```
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipKeyPoints</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>ipDescriptors</td>
<td>Reference To ITcVnImage [383]</td>
<td>Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITcVnImage [383]</td>
<td>Mask to specify, where to look for keypoints (set to 0 if no mask required)</td>
</tr>
<tr>
<td>stParams</td>
<td>Reference To TcVnParamsBRISK [215]</td>
<td>Additional expert parameters</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Matching

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.17.17 F_VN_KeyPointsAndDescriptorsKAZE

Detects keypoints and compute descriptors using the KAZE method.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_KeyPointsAndDescriptorsKAZE : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipKeyPoints : Reference To ITcVnContainer;
    ipDescriptors : Reference To ITcVnImage;
    hrPrev : HRESULT;
END_VAR
```
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [1.383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipKeyPoints</td>
<td>Reference To ITcVnContainer [1.345]</td>
<td>Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>ipDescriptors</td>
<td>Reference To ITcVnImage [1.383]</td>
<td>Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [1.135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [1.135]

### Required License

TC3 Vision Matching

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.17.18 F_VN_KeyPointsAndDescriptorsKAZEExp

Detects keypoints and compute descriptors using the KAZE method. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

### Syntax

**Definition:**

```c
FUNCTION F_VN_KeyPointsAndDescriptorsKAZEExp : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  ipKeyPoints : Reference To ITcVnContainer;
  ipDescriptors : Reference To ITcVnImage;
  ipMask : ITcVnImage;
  stParams : Reference To TcVnParamsKAZE;
  hrPrev : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image</td>
</tr>
<tr>
<td>ipKeyPoints</td>
<td>Reference To ITcVnContainer</td>
<td>Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>ipDescriptors</td>
<td>Reference To ITcVnImage</td>
<td>Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITcVnImage</td>
<td>Mask to specify, where to look for keypoints (set to 0 if no mask required)</td>
</tr>
<tr>
<td>stParams</td>
<td>Reference To TcVnParamsKAZE</td>
<td>Additional expert parameters</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Matching

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.17.19 F_VN_KeyPointsAndDescriptorsORB

**F_VN_KeyPointsAndDescriptorsORB**

ipSrcImage : ITcVnImage  
ipKeyPoints : Reference To ITcVnContainer  
ipDescriptors : Reference To ITcVnImage  
hrPrev : HRESULT

**F_VN_KeyPointsAndDescriptorsORB**

Detects keypoints and compute descriptors using the ORB method.
Can use available TwinCAT Job Tasks for executing parallel code regions.

## Syntax

**Definition:**

```
FUNCTION F_VN_KeyPointsAndDescriptorsORB : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipKeyPoints : Reference To ITcVnContainer;
    ipDescriptors : Reference To ITcVnImage;
    hrPrev : HRESULT;
END_VAR
```
API reference

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image</td>
</tr>
<tr>
<td>ipKeyPoints</td>
<td>Reference To ITcVnContainer</td>
<td>Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>ipDescriptors</td>
<td>Reference To ITcVnImage</td>
<td>Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Matching

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.17.20  F_VN_KeyPointsAndDescriptorsORBExp

Detect keypoints and compute descriptors using the ORB method. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

FUNCTION F_VN_KeyPointsAndDescriptorsORBExp : HRESULT
VAR_INPUT
  ipSrcImage   : ITcVnImage;
  ipKeyPoints  : Reference To ITcVnContainer;
  ipDescriptors: Reference To ITcVnImage;
  ipMask       : ITcVnImage;
  stParams     : Reference To TcVnParamsORB;
  hrPrev       : HRESULT;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>[ITcVnImage]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipKeyPoints</td>
<td>Reference To [ITcVnContainer]</td>
<td>Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>ipDescriptors</td>
<td>Reference To [ITcVnImage]</td>
<td>Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)</td>
</tr>
<tr>
<td>ipMask</td>
<td>[ITcVnImage]</td>
<td>Mask to specify, where to look for keypoints (set to 0 if no mask required)</td>
</tr>
<tr>
<td>stParams</td>
<td>Reference To TcVnParamsORB</td>
<td>Additional expert parameters</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Matching

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### Syntax

**F_VN_KeypointsFAST**

```
FUNCTION F_VN_KeypointsFAST : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipKeyPoints : Reference To ITcVnContainer;
    hrPrev : HRESULT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image (single-channel).</td>
</tr>
<tr>
<td>ipKeyPoints</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Matching

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.17.22 F_VN_KeyPointsFASTExp

Detects keypoints using the FAST method. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

**Syntax**

**Definition:**

FUNCTION F_VN_KeyPointsFASTExp : HRESULT
VAR_INPUT
  ipSrcImage     : ITcVnImage;
  ipKeyPoints    : Reference To ITcVnContainer;
  ipMask         : ITcVnImage;
  stParams       : Reference To TcVnParamsFAST;
  hrPrev         : HRESULT;
END_VAR
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [135]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipKeyPoints</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITcVnImage [383]</td>
<td>Mask to specify, where to look for keypoints (set to 0 if no mask required)</td>
</tr>
<tr>
<td>stParams</td>
<td>Reference To TcVnParamsFAST [216]</td>
<td>Additional expert parameters</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Matching

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.17.23 F_VN_KeyPointsGFTT

F_VN_KeyPointsGFTT

- ipSrcImage  ITCvnImage
- ipKeyPoints Reference To ITcVnContainer
- hrPrev      HRESULT

Detects keypoints using the GFTT method, which detects strong corners. Can use available TwinCAT Job Tasks for executing parallel code regions.

### Syntax

Definition:

FUNCTION F_VN_KeyPointsGFTT : HRESULT

VAR_INPUT
   ipSrcImage    : ITCvnImage;
   ipKeyPoints   : Reference To ITcVnContainer;
   hrPrev        : HRESULT;
END_VAR
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image</td>
</tr>
<tr>
<td>ipKeyPoints</td>
<td>Reference To ITcVnContainer</td>
<td>Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Matching

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.17.24 **F_VN_KeyPointsGFTTExp**

Detects keypoints using the GFTT method, which detects strong corners. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

**Syntax**

**Definition:**

FUNCTION F_VN_KeyPointsGFTTExp : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipKeyPoints : Reference To ITcVnContainer;
    ipMask : ITcVnImage;
    stParams : Reference To TcVnParamsGFTT;
    hrPrev : HRESULT;
END_VAR
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipKeyPoints</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITcVnImage [383]</td>
<td>Mask to specify, where to look for keypoints (set to 0 if no mask required)</td>
</tr>
<tr>
<td>stParams</td>
<td>Reference To TcVnParamsGFTT [216]</td>
<td>Additional expert parameters</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Matching

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.17.25 F_VN_KeyPointsMSER

F_VN_KeyPointsMSER

ipSrcImage ITcVnImage hrPrev HRESULT F_VN_KeyPointsMSER

ipKeyPoints Reference To ITcVnContainer

Detects keypoints using the MSER method.

Syntax

Definition:

FUNCTION F_VN_KeyPointsMSER : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  ipKeyPoints : Reference To ITcVnContainer;
  hrPrev : HRESULT;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipKeyPoints</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Matching

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.17.26 F_VN_KeyPointsMSERExp

F_VN_KeyPointsMSERExp

```
FUNCTION F_VN_KeyPointsMSERExp : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipKeyPoints : Reference To ITcVnContainer;
    ipMask : ITcVnImage;
    stParams : Reference To TcVnParamsMSER;
    hrPrev : HRESULT;
END_VAR
```

Detects keypoints using the MSER method. (expert function)

## Syntax

**Definition:**

FUNCTION F_VN_KeyPointsMSERExp : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipKeyPoints : Reference To ITcVnContainer;
    ipMask : ITcVnImage;
    stParams : Reference To TcVnParamsMSER;
    hrPrev : HRESULT;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipKeyPoints</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITcVnImage [383]</td>
<td>Mask to specify, where to look for keypoints (set to 0 if no mask required)</td>
</tr>
<tr>
<td>stParams</td>
<td>Reference To TcVnParamsMSER [218]</td>
<td>Additional expert parameters</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Matching

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.17.27 F_VN_KeyPointsSB

Detects keypoints using a Simple Blob method: - several iterations apply different thresholds to source image - connected components (blobs) are detected - the center and radius of the blobs are returned as keypoints

Can use available TwinCAT Job Tasks for executing parallel code regions.

### Syntax

**Definition:**

```c
FUNCTION F_VN_KeyPointsSB : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  ipKeyPoints : Reference To ITcVnContainer;
  hrPrev : HRESULT;
END_VAR
```
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipKeyPoints</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Matching

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.17.28 F_VN_KeyPointsSBExp

Detects keypoints using a Simple Blob method (expert function) - several iterations apply different thresholds to source image - connected components (blobs) are detected - the center and radius of the blobs are returned as keypoints

Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```c
FUNCTION F_VN_KeyPointsSBExp : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  ipKeyPoints : Reference To ITcVnContainer;
  ipMask : ITcVnImage;
  stParams : Reference To TcVnParamsSB;
  hrPrev : HRESULT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipKeyPoints</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITcVnImage [383]</td>
<td>Mask to specify, where to look for keypoints (currently not supported, set to 0)</td>
</tr>
<tr>
<td>stParams</td>
<td>Reference To TcVnParamsSB [219]</td>
<td>Several parameters to filter the detected blobs</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Matching

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## Syntax

**F_VN_MatchDescriptorsBF**

```c
FUNCTION F_VN_MatchDescriptorsBF : HRESULT
VAR_INPUT
    ipQueryDescriptors : ITcVnImage;
    ipTrainDescriptors : ITcVnImage;
    ipMatches         : Reference To ITcVnContainer;
    hrPrev            : HRESULT;
END_VAR
```

Match descriptors using a brute force approach.
Can use available TwinCAT Job Tasks for executing parallel code regions.

## Definition:

FUNCTION F_VN_MatchDescriptorsBF : HRESULT
VAR_INPUT
    ipQueryDescriptors : ITcVnImage;
    ipTrainDescriptors : ITcVnImage;
    ipMatches         : Reference To ITcVnContainer;
    hrPrev            : HRESULT;
END_VAR
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipQueryDescriptors</td>
<td>ITcVnImage [383]</td>
<td>Query descriptors</td>
</tr>
<tr>
<td>ipTrainDescriptors</td>
<td>ITcVnImage [383]</td>
<td>Training descriptors</td>
</tr>
<tr>
<td>ipMatches</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container which is filled with the descriptor matches</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(ContainerType_Vector_TcVnDMatch; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Matching

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.17.30 F_VN_MatchDescriptorsBFExp

Match descriptors using a brute force approach. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```c
FUNCTION F_VN_MatchDescriptorsBFExp : HRESULT
VAR_INPUT
    ipQueryDescriptors : ITcVnImage;
    ipTrainDescriptors : ITcVnImage;
    ipMatches : Reference To ITcVnContainer;
    ipMask : ITcVnImage;
    eNormType : ETcVnNormalizationType;
    bCrossCheck : BOOL;
    hrPrev : HRESULT;
END_VAR
```
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipQueryDescriptors</td>
<td>ITcVnImage [383]</td>
<td>Query descriptors</td>
</tr>
<tr>
<td>ipTrainDescriptors</td>
<td>ITcVnImage [383]</td>
<td>Training descriptors</td>
</tr>
<tr>
<td>ipMatches</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container which is filled with the descriptor matches</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITcVnImage [383]</td>
<td>Mask to specify permissible matches, i.e. query[i] can be matched with train[j] only if mask[i][j] != 0 (set to 0 if no mask required)</td>
</tr>
<tr>
<td>eNormType</td>
<td>EITcVnNormalizationType [195]</td>
<td>Normalization type (only L1, L2, L2SQR, HAMMING, HAMMING2 supported). HAMMING2 should be used for ORB descriptors if the ORB nBriefPoints parameter is 3 or 4</td>
</tr>
<tr>
<td>bCrossCheck</td>
<td>BOOL</td>
<td>If true, only consistent matches are returned, i.e. query-&gt;train and train-&gt;query detect the same match</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Matching

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### Syntax

**6.4.17.31 F_VN_MatchDescriptorsFlannLsh**

```plaintext
FUNCTION F_VN_MatchDescriptorsFlannLsh : HRESULT
VAR_INPUT
    ipQueryDescriptors : ITcVnImage;
    ipTrainDescriptors : ITcVnImage;
    ipMatches : Reference To ITcVnContainer;
    hrPrev : HRESULT;
END_VAR
```

**F_VN_MatchDescriptorsFlannLsh**

Match descriptors using a FLANN based approach with LSH index.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_MatchDescriptorsFlannLsh : HRESULT
VAR_INPUT
    ipQueryDescriptors : ITcVnImage;
    ipTrainDescriptors : ITcVnImage;
    ipMatches : Reference To ITcVnContainer;
    hrPrev : HRESULT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipQueryDescriptors</td>
<td>ITcVnImage [IP.383]</td>
<td>Query descriptors</td>
</tr>
<tr>
<td>ipTrainDescriptors</td>
<td>ITcVnImage [IP.383]</td>
<td>Training descriptors</td>
</tr>
<tr>
<td>ipMatches</td>
<td>Reference To ITcVnContainer [IP.345]</td>
<td>Returns a container which is filled with the descriptor matches (ContainerType_Vector_TcVnDMatch; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [IP.135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [IP.135]

**Further information**

- **Number of matches**
  For reasons related to the method, the number of matches can vary from call to call with this function.

**Required License**

TC3 Vision Matching

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**Syntax**

**F_VN_MatchDescriptorsFlannLshExp**

```
FUNCTION F_VN_MatchDescriptorsFlannLshExp : HRESULT
VAR_INPUT
    ipQueryDescriptors : ITcVnImage;
    ipTrainDescriptors : ITcVnImage;
    ipMatches : Reference To ITcVnContainer;
    nTableNumber : UDINT;
    nKeySize : UDINT;
    nMultiProbeLevel : UDINT;
    nChecks : UDINT;
    hrPrev : HRESULT;
RETURN HRESULT F_VN_MatchDescriptorsFlannLshExp;
```

Match descriptors using a FLANN based approach with LSH index. (expert function)
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipQueryDescriptors</td>
<td>ITcVnImage</td>
<td>Query descriptors</td>
</tr>
<tr>
<td>ipTrainDescriptors</td>
<td>ITcVnImage</td>
<td>Training descriptors</td>
</tr>
<tr>
<td>ipMatches</td>
<td>Reference To ITcVnContainer</td>
<td>Returns a container which is filled with the descriptor matches (ContainerType_Vector_TcVnDMatch; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>nTableNumber</td>
<td>UDINT</td>
<td>Number of tables</td>
</tr>
<tr>
<td>nKeySize</td>
<td>UDINT</td>
<td>Key size</td>
</tr>
<tr>
<td>nMultiProbeLevel</td>
<td>UDINT</td>
<td>Multi-probe level</td>
</tr>
<tr>
<td>nChecks</td>
<td>UDINT</td>
<td>Maximum number of visited leaves when searching for neighbors</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

**HRESULT [135]**

### Required License

TC3 Vision Matching

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.17.33 **F_VN_MatchDescriptorsKnnBF**

Match descriptors (k nearest neighbors) using a brute force approach. Can use available TwinCAT Job Tasks for executing parallel code regions.

### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_MatchDescriptorsKnnBF : HRESULT
VAR_INPUT
     ipQueryDescriptors : ITcVnImage;
     ipTrainDescriptors : ITcVnImage;
     ipMatches : Reference To ITcVnContainer;
     nK : UDINT;
     hrPrev : HRESULT;
END_VAR
```

---

[Image of the document page]

**BECKHOFF**

API reference

**TF7000 - TF7300**

Version: 1.3

1209
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipQueryDescriptors</td>
<td>ITcVnImage [383]</td>
<td>Query descriptors</td>
</tr>
<tr>
<td>ipTrainDescriptors</td>
<td>ITcVnImage [383]</td>
<td>Training descriptors</td>
</tr>
<tr>
<td>ipMatches</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container which is filled with the descriptor matches</td>
</tr>
<tr>
<td>nK</td>
<td>UDINT</td>
<td>Number of required best train matches for each query descriptor (i.e. nk := 2)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Matching

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.17.34 **F_VN_MatchDescriptorsKnnBFExp**

```
F_VN_MatchDescriptorsKnnBFExp
```

Match descriptors (k nearest neighbors) using a brute force approach. (expert function)

Can use available TwinCAT Job Tasks for executing parallel code regions.

### Syntax

**Definition:**

```
FUNCTION F_VN_MatchDescriptorsKnnBFExp : HRESULT
VAR_INPUT
  ipQueryDescriptors : ITcVnImage;
  ipTrainDescriptors : ITcVnImage;
  ipMatches : Reference To ITcVnContainer;
  nK : UDINT;
  ipMask : ITcVnImage;
  bCrossCheck : BOOL
  eNormType : ETcnNormalizationType
  bCompactResult : BOOL
```

...
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipQueryDescriptors</td>
<td>ITcVnImage [383]</td>
<td>Query descriptors</td>
</tr>
<tr>
<td>ipTrainDescriptors</td>
<td>ITcVnImage [383]</td>
<td>Training descriptors</td>
</tr>
<tr>
<td>ipMatches</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container which is filled with the descriptor matches (ContainerType_Vector_Vector_TcVnDMatch; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>nK</td>
<td>UDINT</td>
<td>Number of required best train matches for each query descriptor (i.e. nk := 2)</td>
</tr>
<tr>
<td>ipMask</td>
<td>ITcVnImage [383]</td>
<td>Mask to specify permissible matches, i.e. query[i] can be matched with train[j] only if mask[i][j] != 0 (set to 0 if no mask required)</td>
</tr>
<tr>
<td>bCompactResult</td>
<td>BOOL</td>
<td>If true, matches vector does not contain entries for fully masked-out query descriptors</td>
</tr>
<tr>
<td>eNormType</td>
<td>ETcVnNormalizationType [195]</td>
<td>Normalization type (only L1, L2, L2SQR, HAMMING, HAMMING2 supported). HAMMING2 should be used for ORB descriptors if the ORB nBriefPoints parameter is 3 or 4)</td>
</tr>
<tr>
<td>bCrossCheck</td>
<td>BOOL</td>
<td>If true, only consistent matches are returned, i.e. query-&gt;train and train-&gt;query detect the same match (only used if nK = 1)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Matching

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.17.35  F_VN_MatchDescriptorsKnnFlannLsh

Match descriptors (k nearest neighbors) using a FLANN based approach with LSH index.
**Syntax**

**Definition:**

```c
FUNCTION F_VN_MatchDescriptorsKnnFlannLsh : HRESULT
VAR_INPUT
    ipQueryDescriptors : ITcVnImage;
    ipTrainDescriptors : ITcVnImage;
    ipMatches : Reference To ITcVnContainer;
    nK : UDINT;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipQueryDescriptors</td>
<td>ITcVnImage [383]</td>
<td>Query descriptors</td>
</tr>
<tr>
<td>ipTrainDescriptors</td>
<td>ITcVnImage [383]</td>
<td>Training descriptors</td>
</tr>
<tr>
<td>ipMatches</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container which is filled with the descriptor matches (ContainerType_Vector_Vector_TcVnDMatch; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>nK</td>
<td>UDINT</td>
<td>Number of required best train matches for each query descriptor (i.e. nk := 2)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Matching

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.17.36 F_VN_MatchDescriptorsKnnFlannLshExp**

Match descriptors (k nearest neighbors) using a FLANN based approach with LSH index. (expert function)

**Syntax**

**Definition:**

```c
FUNCTION F_VN_MatchDescriptorsKnnFlannLshExp : HRESULT
VAR_INPUT
    ipQueryDescriptors : ITcVnImage;
    ipTrainDescriptors : ITcVnImage;
    ipMatches : Reference To ITcVnContainer;
    nK : UDINT;
    nTableNumber : UDINT;
    nKeySize : UDINT;
    nMultiProbeLevel : UDINT;
    nChecks : UDINT;
    hrPrev : HRESULT;
END_VAR
```
FUNCTION F_VN_MatchDescriptorsKnnFlannLshExp : HRESULT
VAR_INPUT
  ipQueryDescriptors : ITcVnImage;
  ipTrainDescriptors : ITcVnImage;
  ipMatches          : Reference To ITcVnContainer;
  nK                  : UDINT;
  nTableNumber       : UDINT;
  nKeySize           : UDINT;
  nMultiProbeLevel   : UDINT;
  nChecks            : UDINT;
  hrPrev             : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipQueryDescriptors</td>
<td>ITcVnImage [383]</td>
<td>Query descriptors</td>
</tr>
<tr>
<td>ipTrainDescriptors</td>
<td>ITcVnImage [383]</td>
<td>Training descriptors</td>
</tr>
<tr>
<td>ipMatches</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container which is filled with the descriptor matches (ContainerType_Vector_Vector_TcVnDMatch; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>nK</td>
<td>UDINT</td>
<td>Number of required best train matches for each query descriptor (i.e. nk := 2)</td>
</tr>
<tr>
<td>nTableNumber</td>
<td>UDINT</td>
<td>Number of tables</td>
</tr>
<tr>
<td>nKeySize</td>
<td>UDINT</td>
<td>Key size</td>
</tr>
<tr>
<td>nMultiProbeLevel</td>
<td>UDINT</td>
<td>Multi-probe level</td>
</tr>
<tr>
<td>nChecks</td>
<td>UDINT</td>
<td>Maximum number of visited leafs when searching for neighbors</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Matching

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.17.37 F_VN_RegionsMSER

F_VN_RegionsMSER

ipSrcImage ITcVnImage
ipRegions Reference To ITcVnContainer
ipBoundingBoxes Reference To ITcVnContainer
hrPrev HRESULT

Detects regions using the MSER method.
**Syntax**

**Definition:**

FUNCTION FgetLocation : HRESULT
VAR_INPUT
   ipSrcImage : ITcVnImage;
   ipRegions : Reference To ITcVnContainer;
   ipBoundingBoxes : Reference To ITcVnContainer;
   hrPrev : HRESULT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipRegions</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container which is filled with the region points (ContainerType_Vector_Vector_TcVnPoint2_DINT; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>ipBoundingBoxes</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container which is filled with the region bounding boxes (ContainerType_Vector_TcVnRectangle_DINT; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Matching

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.17.38  F_VN_RegionsMSERExp**

Detects regions using the MSER method. (expert function)

**Syntax**

**Definition:**

FUNCTION F_VN_RegionsMSERExp : HRESULT
VAR_INPUT
   ipSrcImage : ITcVnImage;
   ipRegions : Reference To ITcVnContainer;
   ipBoundingBoxes : Reference To ITcVnContainer;
   stParams : Reference To TcVnParamMSER;
   hrPrev : HRESULT;
END_VAR
stParams : Reference To TcVnParamsMSER;
hrPrev : HRESULT;

## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image</td>
</tr>
<tr>
<td>ipRegions</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container which is filled with the region points (ContainerType_Vector_Vector_TcVnPoint2_DINT; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>ipBoundingBoxes</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns a container which is filled with the region bounding boxes (ContainerType_Vector_TcVnRectangle_DINT; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>stParams</td>
<td>Reference To TcVnParamsMSER [218]</td>
<td>Additional expert parameters</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Matching

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

## 6.4.18 Measurement

The group contains functions for the measurement of objects in images.

### Functions

#### Edge localization

- **F_VN_LocateCircularArc(Exp) [1227]** for localizing proportional circular arcs
- **F_VN_LocateEdge(Exp) [1233]** for localizing linear edges
- **F_VN_LocateEdges(Exp) [1240]** for localizing several linear edges
- **F_VN_LocateEllipse(Exp) [1244]** for localizing complete ellipses

#### Geometrical features

- **F_VN_ClosestPointsBF [1222]** for calculating the closest points in two point clouds
- **F_VN_MeasureAngleBetweenEdges(Exp) [1250]** for measuring the angle between two edges
- **F_VN_MeasureEdgeDistance(Exp) [1255]** for measuring the average distance between two edges
- **F_VN_MeasureMinEdgeDistance(Exp) [1260]** for measuring the minimum distance between two edges

#### Miscellaneous

- **F_VN_AdjustSearchWindowOrientationToLinearEdge [1221]** for aligning the search field to an edge
Parameters for edge localization

The borders of an image object often don't lie exactly on the border between two pixels. Instead, they run independently of pixel grids. Therefore edge localization with sub-pixel accuracy is necessary. This not only considers the transition between two pixels, but also determines the edge position within a pixel on the basis of the overall intensity gradient of an object border.

NOTE

Don't process the image before measuring

Don't process the image before calling the measurement function. Processing can lead to loss of information and lower accuracy accordingly.

This basic functionality of edge localization is used by all measurement functions. The functions differ mainly by the geometric arrangement in which they perform the edge localization (e.g. F_VN_LocateEllipse for round objects) and the properties that you calculate from them (e.g. F_VN_MeasureAngleBetweenEdges for angle measurement).

Certain parameters must be defined for each function that uses edge localization. These include:

- Definition of the search area: where is the edge sought?
- Definition of edge contrast: What is detected as an edge?
- Definition of the measurement algorithm: How to determine the exact position of the edge.

These parameters are described below. In the case of expert parameters, standard values are specified in addition.

Search area
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Position of the search lines</strong></td>
<td>The position of the search lines is defined according to the feature to be localized.</td>
</tr>
<tr>
<td></td>
<td>The position of search lines for a linear edge is defined as a single line with the start point aStartPoint and end point aEndPoint.</td>
</tr>
<tr>
<td></td>
<td>The position of search lines for an ellipse is defined by a center point aCenter and a radius fSearchRadius.</td>
</tr>
<tr>
<td></td>
<td>The position of search lines for angles is defined by an inner point aInnerPoint and two outer points aOuterPoint1 and aOuterPoint2.</td>
</tr>
<tr>
<td></td>
<td>The position of the search lines should always be set such that the search lines cut the edges to be localized orthogonally. A rectangular search area can be adjusted orthogonally to an edge with the function F_VN_AdjustSearchWindowOrientationToLinearEdge.</td>
</tr>
<tr>
<td><strong>Number of search lines</strong></td>
<td>This specifies how many search lines are to be used for the edge localization. The simplest edge localization has one search line on which the edge must be found.</td>
</tr>
<tr>
<td>nSearchLines : UDINT</td>
<td>Depending on the geometric feature to be determined, multiple search lines are distributed locally so that several edge transitions are localized.</td>
</tr>
<tr>
<td></td>
<td>This can be necessary in the following cases:</td>
</tr>
<tr>
<td></td>
<td>• Not only the position, but also the gradient or the rotation of the edge are required.</td>
</tr>
<tr>
<td></td>
<td>• The geometrical feature (e.g. ellipse) necessitates several search lines.</td>
</tr>
<tr>
<td></td>
<td>• An average value is required.</td>
</tr>
<tr>
<td></td>
<td>When localizing linear edges, the search lines are arranged symmetrically on both sides of the central search line, the central search line being included in nSearchLines. Thus, an odd number of search lines is always required.</td>
</tr>
<tr>
<td><strong>Distance between the search lines</strong></td>
<td>The search lines distance defines the distance between neighboring search lines in pixels (&gt; 0, default 1). This allows flexible balancing between accuracy and computing time.</td>
</tr>
<tr>
<td>fSearchlineDist : REAL</td>
<td>(Expert)</td>
</tr>
<tr>
<td></td>
<td>Default: 1</td>
</tr>
<tr>
<td><strong>Inversion of the search direction</strong></td>
<td>The localization algorithm always searches for the edge from one end of the search line to the other. E.g. searching takes place from aStartPoint to aEndPoint. If it is necessary to invert this search direction due to the image properties (i.e. to start from the other end of the search lines), this parameter helps.</td>
</tr>
<tr>
<td>bInvertSearchDirection : BOOL</td>
<td>(Expert)</td>
</tr>
<tr>
<td></td>
<td>Default: FALSE</td>
</tr>
</tbody>
</table>

**Edge contrast**
### Parameter | Description
--- | ---
**Edge direction**
```
eEdgeDirection : ETCVnEdgeDirection
```
The edge direction determines whether an edge transition from light to dark or from dark to light should be found. The following values are available for this:
- TCVN_ED_DARK_TO_LIGHT
- TCVN_ED_LIGHT_TO_DARK

**Minimum strength**
```
fMinStrength : LREAL
```
The minimum strength defines the minimum difference in intensity required within an edge for it to be found. This makes it possible to control how much contrast an edge must have in order to be found.

**Maximum thickness**
```
nMaxThickness : UDINT
```
*(Expert)*

**Default:** 10

The maximum thickness specifies the number of pixels within which `fMinStrength` must be reached. This allows you to control whether only sharp edges or also very blurred edges are to be found.

**Algorithm**
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Algorithm</strong></td>
<td>This parameter defines the localization algorithm used. The available options are:</td>
</tr>
<tr>
<td>eAlgorithm : ETCvNEdgeDetectionAlgorithm</td>
<td>• TCVN_EDA_INTERPOLATION&lt;br&gt;• TCVN_EDA_APPROX_ERF&lt;br&gt;• TCVN_EDA_APPROX_GAUSSIAN&lt;br&gt;TCVN_EDA_APPROX_ERF and TCVN_EDA_APPROX_GAUSSIAN are slower than the standard algorithm TCVN_EDA_INTERPOLATION, but can achieve higher accuracy. The prerequisite for this, however, is that the intensity gradient of the edge also corresponds as well as possible to the specified model. The error function erf or the Gaussian function is used as a model. The erf function is well suited for edges if the transition from bright to dark extends over several pixels and remains approximately constant on both sides of the light or dark area respectively. A Gaussian function is well suited for lines, i.e. a dark-light-dark transition or vice versa. However, the brightness plateau may only be a few pixels wide, otherwise the erf function should be used.</td>
</tr>
<tr>
<td><strong>Subpixels / Iterations</strong></td>
<td>The meaning of this parameter depends on the localization algorithm used.</td>
</tr>
<tr>
<td>nSubpixelsIterations : UDINT</td>
<td>At TCVN_EDA_INTERPOLATION: the number of subpixels considered defines the fine granularity with which the search line evaluates the underlying pixel intensities. A value of 10, for example, specifies that 10 measured values are taken over a length of 1 pixel. Recommended: 5 - 10 At TCVN_EDA_APPROX_ERF and TCVN_EDA_APPROX_GAUSSIAN: the number of iterations is an abort criterion that defines the maximum number of iterations with which the edge model is to be approximated for the pixel intensities. More iterations increase the runtime; if this is too high the interpolation method should be used. Recommended: 50 - 100</td>
</tr>
<tr>
<td><strong>Approximation accuracy</strong></td>
<td>The approximation accuracy is a further abort criterion of the edge localization. It is only observed if the localization algorithm is TCVN_EDA_APPROX_ERF or TCVN_EDA_APPROX_GAUSSIAN. The approximation of the edge or line is aborted if either the number of iterations of nSubPixelIterations is reached or the relative</td>
</tr>
<tr>
<td>fApproxPrecision : REAL</td>
<td></td>
</tr>
</tbody>
</table>
### Interpretation of the HRESULT

The HRESULT [135] return values have the following meaning:

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>16#000</td>
<td>S_OK</td>
<td>Function was successfully executed and an edge was located on all search lines.</td>
</tr>
<tr>
<td>16#001</td>
<td>S_FALSE</td>
<td>Function was successfully executed, but edges were not located on each search line.</td>
</tr>
<tr>
<td>16#256</td>
<td>S_WATCHDOGTIMEROUT</td>
<td>Function was aborted by the watchdog. In this case, all measurement functions can return partial results.</td>
</tr>
<tr>
<td>16#70C</td>
<td>NOTFOUND</td>
<td>Function was executed successfully, but no edge could be found on any search line. It is assumed that a measurement function is only called if it is ensured that the image contains an object with the parameterized edge properties. If none of the search lines are found, this indicates incorrect parameterization or an unsuitable call of the function. Therefore, an error code is output in this case.</td>
</tr>
<tr>
<td>16#7xx</td>
<td>All error codes except NOTFOUND</td>
<td>Function was not executed successfully. See also: ADS Return Codes [1486]</td>
</tr>
</tbody>
</table>

The following query, for example, can be used for the processing of the return value:

```vbnet
IF SUCCEEDED(hr) THEN
  // Process results
  IF hr <> S_OK THEN
    // Emit warning that not all search lines were found
  END_IF
ELSIF hr = Tc2_System.E_RESULTAdsErr.NOTFOUND THEN
  // Emit warning for wrong parametrization or missing piece
ELSE
  // Error handling
END_IF
```

### Reducing execution time

The execution time for a fixed search window varies slightly depending on the position of the component in the image. If the edge that is found is closer to the starting point, the execution time is shorter than for an edge further away.

A larger fluctuation can occur with the two approximation algorithms, since these require different iterations in order to approximate the model parameters using the surrounding pixel intensities.

In general, fewer iterations are required if the search lines of the search window hit the edge as orthogonally as possible. If the position and orientation of the objects in the image are subject to little or no fluctuation, the search window can be adjusted accordingly to achieve shorter execution times. In addition, the maximum time required can be reduced by reducing the maximum number of iterations, but this may lead to less accurate results.

In addition, the time can be limited with an external watchdog, which terminates the execution of the function if necessary and returns the existing partial results.

### Samples

Measurement [1423]
6.4.18.1  F_VN_AdjustSearchWindowOrientationToLinearEdge

Adjust the search window to be orthogonal to a linear edge. Might be helpful before calling edge localization and distance measurement functions that rely on the search window.

Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_AdjustSearchWindowOrientationToLinearEdge : HRESULT
VAR_INPUT
  ipSrcImage : ITCvImage;
  aStartPoint : Reference To TcVnPoint2_REAL;
  aEndPoint : Reference To TcVnPoint2_REAL;
  eEdgeDirection : ETcVnEdgeDirection;
  fMinStrength : REAL;
  nMaxThickness : UDINT;
  nSearchLines : UDINT;
  hrPrev : HRESULT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvImage</td>
<td>Source image</td>
</tr>
<tr>
<td>aStartPoint</td>
<td>Reference To TcVnPoint2_REAL</td>
<td>Position from which to start the search process (in the direction of aEndPoint), which is not changed</td>
</tr>
<tr>
<td>aEndPoint</td>
<td>Reference To TcVnPoint2_REAL</td>
<td>Position where the search process ends, which is adjusted by this function</td>
</tr>
<tr>
<td>eEdgeDirection</td>
<td>ETcVnEdgeDirection</td>
<td>Specification of the edge direction to search for</td>
</tr>
<tr>
<td>fMinStrength</td>
<td>REAL</td>
<td>Specification of the minimum strength (intensity difference) of the edge to search for</td>
</tr>
<tr>
<td>nMaxThickness</td>
<td>UDINT</td>
<td>Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels</td>
</tr>
<tr>
<td>nSearchLines</td>
<td>UDINT</td>
<td>Width of the search window, i.e. the number of search lines (3, 5, 7, ...), centered around the line specified by aStartPoint and aEndPoint</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Further information

The function F_VN_AdjustSearchWindowOrientationToLinearEdge rotates the search field for measurement functions in such a way that it lies orthogonal to a linear edge.
Parameter

Input image

The input image ipSrcImage must contain an edge in accordance with the parameters for edge detection.

Search field (Return value)

The search field to be adjusted is transferred as a reference so that it can be adjusted. The search field consists of the central search line, which consists of the start point aStartPoint and end point aEndPoint, as well as the number of search lines nSearchLines.

Following the function call, aStartPoint and aEndPoint contain the adjusted central search line. This is now orthogonal to the linear edge on the input image ipSrcImage.

Parameters for edge localization

The parameters eEdgeDirection, fMinStrength and nMaxThickness determine the search features for the linear edge in the input image ipSrcImage. These work analogously to the edge localization in F_VN_LocateEdge [ 1230 ].

Required License

TC3 Vision Metrology 2D

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.18.2 F_VN_ClosestPointsBF

Find the closest distance between two 2d point sets of the same type, using a brute force approach.

Syntax

Definition:

```
FUNCTION F_VN_ClosestPointsBF : HRESULT
VAR_INPUT
    ipPointSet1 : ITcVnContainer;
    ipPointSet2 : ITcVnContainer;
    fMinDist    : Reference To LREAL;
    aPoint1     : Reference To TcVnPoint2_LREAL;
    aPoint2     : Reference To TcVnPoint2_LREAL;
    hrPrev      : HRESULT;
END_VAR
```

Find the closest distance between two 2d point sets of the same type, using a brute force approach.
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipPointSet1</td>
<td>ITcVnContainer [345]</td>
<td>Container with 1st set of 2d points</td>
</tr>
<tr>
<td>ipPointSet2</td>
<td>ITcVnContainer [345]</td>
<td>Container with 2nd set of 2d points</td>
</tr>
<tr>
<td>fMinDist</td>
<td>Reference To LREAL</td>
<td>Returns the minimum distance between 2 points of the different point sets</td>
</tr>
<tr>
<td>aPoint1</td>
<td>Reference To TcVnPoint2_LREAL [151]</td>
<td>Returns the point out of ipPointSet1, for which fMinDist is achieved</td>
</tr>
<tr>
<td>aPoint2</td>
<td>Reference To TcVnPoint2_LREAL [151]</td>
<td>Returns the point out of ipPointSet2, for which fMinDist is achieved</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Further information

The function F_VN_ClosestPointsBF determines which points from two point clouds are closest to each other.

### Parameter

#### Point clouds

The closest two points are sought in the point clouds `ipPointSet1` and `ipPointSet2`.

#### Minimum distance (Return value)

The parameter `fMinDist` returns the distance between the two points that are closest to each other. This corresponds to the minimum distance between the two point clouds.

#### Closest points (Return value)

The parameters `aPoint1` and `aPoint2` return the two points that are closest to each other in the point clouds.

### Application

The determination of the minimum distance between two lines looks like this, for example:

```c
hr := F_VN_ClosestPointsBF(
    ipPointSet1 := ipLine1,
    ipPointSet2 := ipLine2,
    fMinDist := fDistance,
    aPoint1 := aPoint1,
    aPoint2 := aPoint2,
    hrPrev := hr
);
```

### Related functions

- F_VN_ClosestPointsBF [1222] for calculating the closest points in two point clouds
- F_VN_MeasureAngleBetweenEdges(Exp) [1250] for measuring the angle between two edges
- F_VN_MeasureEdgeDistance(Exp) [1255] for measuring the average distance between two edges
- F_VN_MeasureMinEdgeDistance(Exp) [1260] for measuring the minimum distance between two edges
Required License
TC3 Vision Metrology 2D

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.18.3  **F_VN_LocateCircularArc**

Locate a circular arc, using an interpolation method.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

**Syntax**

**Definition:**

```c
definition:  
FUNCTION F_VN_LocateCircularArc : HRESULT  
VAR_INPUT  
ipSrcImage : ITcVnImage;  
stCircularArc : Reference To TcVnCircularArc;  
aCenter : Reference To TcVnPoint2_REAL;  
fSearchRadius : REAL;  
fArcDirectionRad : LREAL;  
eEdgeDirection : ETcVnEdgeDirection;  
fMinStrength : REAL;  
hrPrev : HRESULT;  
END_VAR  
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image (1 channel)</td>
</tr>
<tr>
<td>stCircularArc</td>
<td>Reference To</td>
<td>Returns the circular arc parameters</td>
</tr>
<tr>
<td></td>
<td>TcVnCircularArc</td>
<td></td>
</tr>
<tr>
<td>aCenter</td>
<td>Reference To</td>
<td>Input estimated circle center (only used as a starting point to search for</td>
</tr>
<tr>
<td></td>
<td>TcVnPoint2_REAL</td>
<td>the circle contour, not used for the circle center estimation)</td>
</tr>
<tr>
<td>fSearchRadius</td>
<td>REAL</td>
<td>Input search radius (starting from aCenter, should be greater than the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>actual circle radius but aCenter + fSearchRadius should be within the image</td>
</tr>
<tr>
<td></td>
<td></td>
<td>borders)</td>
</tr>
<tr>
<td>fArcDirectionRad</td>
<td>LREAL</td>
<td>Input search starting direction in radian. Valid range is [-pi, +pi], where</td>
</tr>
<tr>
<td></td>
<td></td>
<td>right is 0 rad, top is -pi/2 rad and bottom +pi/2 rad. The circular arc</td>
</tr>
<tr>
<td></td>
<td></td>
<td>should at least be valid in range fArcDirectionRad + 0.4 rad.</td>
</tr>
<tr>
<td>eEdgeDirection</td>
<td>ETcVnEdgeDirection</td>
<td>Specification of the edge direction to search for</td>
</tr>
<tr>
<td>fMinStrength</td>
<td>REAL</td>
<td>Specification of the minimum strength (intensity difference) of the edge to</td>
</tr>
<tr>
<td></td>
<td></td>
<td>search for</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Further information**

The function `F_VN_LocateCircularArc` localizes and measures a proportional circular arc. It is based on the edge localization [1216] and uses normal parameters. It returns the localized circular arc as a structure of the type `TcVnCircularArc` (center point, radius, angle range).

If full circles are to be measured instead of a circular arc, use the function `F_VN_LocateEllipse` [1242] instead.

**Parameter**

**Input image**

The input image `ipSrcImage` must be a 1-channel grayscale image.

**Circular arc** (Return value)

The parameter `stCircularArc` returns the localized circular arc as a structure of the type `TcVnCircularArc` [206].

**Estimated center point**

The estimated center point `aCenter` of the circular arc must be specified in order to define the position of the search lines.

If the position of the center point of the circle is unknown, it may need to be determined beforehand by means of an object detection [1027].

**Search radius**

The search radius `fSearchRadius` defines the length of the search lines from the center point of the circle in pixels. The search radius should be defined somewhat larger than the actual radius of the circle so that the edge localization can be carried out unambiguously.
Search direction

The search direction fArcDirectionRad specifies the circular direction in which the edge search should begin (starting from the center point of the circle). The value is to be specified in radians and has a value range of [-π, π]. Right corresponds to 0, top with −π/2 and bottom with π/2. Starting from fArcDirectionRad, the circular arc must be present at least in the range of ± 0.4 in order to be detected. It is recommended to aim for the center of the circular arc.

Parameters for edge localization

The remaining parameters are explained in detail in the chapter Edge localization [1216].

Expert parameters


Application

The localization and measurement of a proportional circular arc with a radius of 270px and a search angle of 120° looks like this, for example:

```pascal
VAR
  stArc : TcVnCircularArc;
  aCenter : TcVnPoint2_REAL := [420, 310];
END_VAR
hr := F_VN_LocateCircularArc(
  ipSrcImage := ipImageIn,
  stCircularArc := stArc,
  aCenterPoint := aCenter,
  fSearchRadius := 270,
  fArcDirectionRad := 2.1, // 2.1 rad equals 120°
  eEdgeDirection := TCVN_ED_LIGHT_TO_DARK,
  fMinStrength := 100,
  hrPrev := hr
);
```

The properties of the localized circular arc can then be retrieved via the variables stArc.aCenter, stArc.fRadius, stArc.fStartAngle and stArc.fEndAngle. In addition, the circular arc can be visualized with the function F_VN_DrawCircularArc [888]:

```pascal
F_VN_DrawCircularArc(stArc, ipImageRes, aColorGreen, 5, hr);
```

Samples

• Locate Circular Arc [1423]

Related functions

• F_VN_LocateCircularArc(Exp) [1227] for localizing proportional circular arcs
• F_VN_LocateEdge(Exp) [1233] for localizing linear edges
• F_VN_LocateEdges(Exp) [1240] for localizing several linear edges
• F_VN_LocateEllipse(Exp) [1244] for localizing complete ellipses

Required License

TC3 Vision Metrology 2D

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.18.4  F_VN_LocateCircularArcExp

Locate a circular arc. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

FUNCTION F_VN_LocateCircularArcExp : HRESULT
VAR_INPUT
    ipSrcImage   : ITcVnImage;
    stCircularArc : Reference To TcVnCircularArc;
    aCenter      : Reference To TcVnPoint2_REAL;
    fSearchRadius : REAL;
    fArcDirectionRad : LREAL;
    eEdgeDirection : ETcVnEdgeDirection;
    fMinStrength  : REAL;
    fAngleStepRad : LREAL;
    nMaxThickness : UDINT;
    bInvertSearchDirection : BOOL;
    nSubpixelsIterations : UDINT;
    fApproxPrecision : REAL;
    eAlgorithm : ETcVnEdgeDetectionAlgorithm;
    ipContourPoints : Reference To ITcVnContainer;
    hrPrev         : HRESULT;
END_VAR
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [1383]</td>
<td>Source image (1 channel)</td>
</tr>
<tr>
<td>stCircularArc</td>
<td>Reference To TcVnCircularArc [206]</td>
<td>Returns the circular arc parameters</td>
</tr>
<tr>
<td>aCenter</td>
<td>Reference To TcVnPoint2_REAL [151]</td>
<td>Input estimated circle center (only used as a starting point to search for the circle contour, not used for the circle center estimation)</td>
</tr>
<tr>
<td>fSearchRadius</td>
<td>REAL</td>
<td>Input search radius (starting from aCenter, should be greater than the actual circle radius but aCenter + fSearchRadius should be within the image borders)</td>
</tr>
<tr>
<td>fArcDirectionRad</td>
<td>LREAL</td>
<td>Input search starting direction in radian. Valid range is [-pi, +pi], where right is 0 rad, top is -pi/2 rad and bottom +pi/2 rad. The circular arc should at least be valid in range fArcDirectionRad + 4 * fAngleStepRad</td>
</tr>
<tr>
<td>eEdgeDirection</td>
<td>ETcVnEdgeDirection [186]</td>
<td>Specification of the edge direction to search for</td>
</tr>
<tr>
<td>fMinStrength</td>
<td>REAL</td>
<td>Specification of the minimum strength (intensity difference) of the edge to search for</td>
</tr>
<tr>
<td>fAngleStepRad</td>
<td>LREAL</td>
<td>Search step in rad (should be chosen so that about 20 - 60 steps are available for the whole arc. In most cases 0.1 rad (5.7 deg) is a good value)</td>
</tr>
<tr>
<td>nMaxThickness</td>
<td>UDINT</td>
<td>Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels</td>
</tr>
<tr>
<td>bInvertSearch</td>
<td>BOOL</td>
<td>If true, the search starts from outside the circular arc in direction of the center</td>
</tr>
<tr>
<td>nSubpixelsIters</td>
<td>UDINT</td>
<td>Specifies the number of subpixels (for INTERPOLATION, 10 - 20 usually is sufficient) or maximum number of iterations for optimizing the parameters (for APPROX_ERF and APPROX_GAUSSIAN, 50 - 100 usually is sufficient)</td>
</tr>
<tr>
<td>fApproxPrecision</td>
<td>REAL</td>
<td>Specifies the approximation precision for APPROX_ERF and APPROX_GAUSSIAN (0.001 usually is sufficient, unused for INTERPOLATION)</td>
</tr>
<tr>
<td>eAlgorithm</td>
<td>ETcVnEdgeDetectionAlgor ithm [185]</td>
<td>Selection of the edge detection algorithm</td>
</tr>
<tr>
<td>ipContourPoints</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns the subpixel accurate contour (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2_REAL; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Further information

The function F_VN_LocateCircularArcExp is the expert version of F_VN_LocateCircularArc [1224]. It contains additional parameters.

Parameter

Input image
The input image \( \text{ipSrcImage} \) must be a 1-channel grayscale image.

**Circular arc** (Return value)

The parameter \( \text{stCircularArc} \) returns the localized circular arc as a structure of the type \( \text{TcVnCircularArc} \) [206].

**Estimated center point**

The estimated center point \( \text{aCenter} \) of the circular arc must be specified in order to define the position of the search lines.

If the position of the center point of the circle is unknown, it may need to be determined beforehand by means of an object detection [1027].

**Search radius**

The search radius \( \text{fSearchRadius} \) defines the length of the search lines from the center point of the circle in pixels. The search radius should be defined somewhat larger than the actual radius of the circle so that the edge localization can be carried out unambiguously.

**Search direction**

The search direction \( \text{fArcDirectionRad} \) specifies the circular direction in which the edge search should begin (starting from the center point of the circle). The value is to be specified in radians and has a value range of \([-\pi, \pi]\). Right corresponds to 0, top with \(-\pi/2\) and bottom with \(\pi/2\). Starting from \(\text{fArcDirectionRad}\), the circular arc must be present at least in the range of \(\pm 0.4\) in order to be detected. It is recommended to aim for the center of the circular arc.

**Radial search line distance** (Expert)

The radial search line distance \( \text{fAngleStepRad} \) defines how large the angle steps between the search lines are. It is specified in radians. The distance should be selected so that the complete circular arc is divided into about 20 – 60 steps. In most cases \(0.1\ \text{rad} (5.7\degree)\) is a suitable value.

**Search direction inversion** (Expert)

The search direction can be inverted with the Boolean \( \text{bInvertSearchDirection} \). In this case, that means the following:

- **FALSE**: Search from the center point of the circle to outside the circular arc.
- **TRUE**: Search from outside the circular arc to the center point of the circle

**Contour points** (Expert, return value)

The container \( \text{ipContourPoints} \) returns the edge points of the circular arc found and is therefore of the type \( \text{ContainerType_Vector_TcVnPoint2_REAL} \). The container contains one point for each search line successfully found.

The parameter can be set to 0 if the return of the edge points is not required.

**Circular arc \( \text{stCircularArc} \)**

The circular arc \( \text{stCircularArc} \) is approximated from the set of these points.

**Parameters for edge localization**

The remaining parameters are explained in detail in the chapter **Edge localization** [1216].

**Application**

The localization of a circular arc with expert parameters looks like this, for example:

```c
hr := F_VN_LocateCircularArc(
    \text{ipSrcImage} := \text{ipImageIn},
    \text{stCircularArc} := \text{stArc},
    \text{aCenterPoint} := \text{aCenter},
```
The localized points can be visualized with the function `F_VN_DrawContours`:

```
hr := F_VN_DrawContours(ipEllipseContour, -1, ipImageRes, aColorGreen, 5, hr);
```

### Samples

- **Locate Circular Arc [1423]**

### Related functions

- `F_VN_LocateCircularArc(Exp)` [1227] for localizing proportional circular arcs
- `F_VN_LocateEdge(Exp)` [1233] for localizing linear edges
- `F_VN_LocateEdges(Exp)` [1240] for localizing several linear edges
- `F_VN_LocateEllipse(Exp)` [1244] for localizing complete ellipses

### Required License

TC3 Vision Metrology 2D

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.18.5 F_VN_LocateEdge

The points of the first occurring edge inside a specified search window, using an interpolation method.

Can use available TwinCAT Job Tasks for executing parallel code regions.

Can return partial results when canceled by Watchdog.

**Syntax**

**Definition:**

```
FUNCTION F_VN_LocateEdge : HRESULT
VAR_INPUT
  ipSrcImage : ITCvImage;
  ipEdgePoints : Reference To ITCvContainer;
  aStartPoint : Reference To TcVnPoint2_REAL;
  aEndPoint : Reference To TcVnPoint2_REAL;
  eEdgeDirection : TcVnEdgeDirection;
  fMinStrength : REAL;
  nSearchLines : UDINT;
  hrPrev : HRESULT;
```

Locate the points of the first occurring edge inside a specified search window, using an interpolation method.
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image (1 channel)</td>
</tr>
<tr>
<td>ipEdgePoints</td>
<td>Reference To ITcVnContainer</td>
<td>Returns the detected edge points (ContainerType_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>aStartPoint</td>
<td>Reference To TcVnPoint2_REAL</td>
<td>Position from which to start the search process (in the direction of aEndPoint)</td>
</tr>
<tr>
<td>aEndPoint</td>
<td>Reference To TcVnPoint2_REAL</td>
<td>Position where the search process ends</td>
</tr>
<tr>
<td>eEdgeDirection</td>
<td>ETcVnEdgeDirection</td>
<td>Specification of the edge direction to search for</td>
</tr>
<tr>
<td>fMinStrength</td>
<td>REAL</td>
<td>Specification of the minimum strength (intensity difference) of the edge to search for</td>
</tr>
<tr>
<td>nSearchLines</td>
<td>UDINT</td>
<td>Width of the search window, i.e. the number of search lines (1, 3, 5, 7, ...), centered around the line specified by aStartPoint and aEndPoint</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fAvgStrength</td>
<td>REAL</td>
<td>Returns the average strength of the detected edge</td>
</tr>
</tbody>
</table>

Return value

HRESULT

Further information

The function F_VN_LocateEdge localizes an edge in a defined search window.

Parameter

Input image

The input image ipSrcImage must be a 1-channel grayscale image.

Localized points (Return value)

The container ipEdgePoints returns the localized points of the linear edge.

The search window
First, a search window has to be defined to determine where and in which direction to search for an edge. This is done with the parameters aStartPoint, aEndPoint and nSearchLines. The search window results from the search line from aStartPoint to aEndPoint and parallel to it (nSearchLines - 1) / 2 search lines in each direction, each with 1 pixel distance. Thus nSearchLines defines the number of search lines and must always have an odd value.

![Diagram of search window](image)

**Fig. 22:** S: StartPoint, E: EndPoint, W: WindowWidth

**Edge strength** *(Return value)*

The return value fAvgStrength displays the actual contrast of the localized edge (averaged over all search lines).

**Parameters for edge localization**

The remaining parameters are explained in detail in the chapter [Edge localization](#edge-localization).

**Expert parameters**

The expert version F_VN_LocateEdgeExp contains additional parameters.

**Application**

The localization of a linear edge looks like this, for example:

```plaintext
hr := F_VN_LocateEdge(
    ipSrcImage := ipImageIn,
    ipEdgePoints := ipEdgePoints,
    aStartPoint := aStartPoint,
    aEndPoint := aEndPoint,
    eEdgeDirection := eDirection,
    fMinStrength := fMinStrength,
    nSearchLines := nSearchLines,
    hrPrev := hr,
    fAvgStrength => fAvgStrength
);
```

The localized edge points can be visualized with the function F_VN_DrawContours.[890]

```plaintext
F_VN_DrawContours(ipEdgePoints, -1, ipImageRes, aColorGreen, 5, hr);
```

**Samples**

- Locate Edge[1427]
Related functions

- `F_VN_LocateCircularArc(Exp)` [1227] for localizing proportional circular arcs
- `F_VN_LocateEdge(Exp)` [1233] for localizing linear edges
- `F_VN_LocateEdges(Exp)` [1240] for localizing several linear edges
- `F_VN_LocateEllipse(Exp)` [1244] for localizing complete ellipses

Required License

TC3 Vision Metrology 2D

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.18.6  `F_VN_LocateEdgeExp`

```
FUNCTION F_VN_LocateEdgeExp : HRESULT
VAR_INPUT
  ipSrcImage         : ITCnImage;
  ipEdgePoints       : Reference To ITCnContainer;
  aStartPoint        : Reference To TcVnPoint2_REAL;
  aEndPoint          : Reference To TcVnPoint2_REAL;
  eEdgeDirection     : ETcVnEdgeDetectionAlgorithm;
  fMinStrength       : REAL;
  nSearchLines       : UDINT;
  fSearchLineDist    : REAL;
  nMaxThickness      : UDINT;
  nSubpixelsIterations : UDINT;
  fApproxPrecision   : REAL;
  eAlgorithm         : ETcVnEdgeDetectionAlgorithm;
  hrPrev             : HRESULT;
END_VAR
VAR_OUTPUT
  fAvgStrength       : REAL;
END_VAR
```

Locate the points of the first occurring edge inside a specified search window with subpixel accuracy. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_LocateEdgeExp : HRESULT
VAR_INPUT
  ipSrcImage         : ITCnImage;
  ipEdgePoints       : Reference To ITCnContainer;
  aStartPoint        : Reference To TcVnPoint2_REAL;
  aEndPoint          : Reference To TcVnPoint2_REAL;
  eEdgeDirection     : ETcVnEdgeDetectionAlgorithm;
  fMinStrength       : REAL;
  nSearchLines       : UDINT;
  fSearchLineDist    : REAL;
  nMaxThickness      : UDINT;
  nSubpixelsIterations : UDINT;
  fApproxPrecision   : REAL;
  eAlgorithm         : ETcVnEdgeDetectionAlgorithm;
  hrPrev             : HRESULT;
END_VAR
VAR_OUTPUT
  fAvgStrength       : REAL;
END_VAR
```
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvImage [383]</td>
<td>Source image (1 channel)</td>
</tr>
<tr>
<td>ipEdgePoints</td>
<td>Reference To ITCvImageContainer [345]</td>
<td>Returns the detected edge points</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(ContainerType_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>aStartPoint</td>
<td>Reference To TcVnPoint2_REAL [151]</td>
<td>Position from which to start the search process (in the direction of aEndPoint)</td>
</tr>
<tr>
<td>aEndPoint</td>
<td>Reference To TcVnPoint2_REAL [151]</td>
<td>Position where the search process ends</td>
</tr>
<tr>
<td>eEdgeDirection</td>
<td>ETcVnEdgeDirection [186]</td>
<td>Specification of the edge direction to search for</td>
</tr>
<tr>
<td>fMinStrength</td>
<td>REAL</td>
<td>Specification of the minimum strength (intensity difference) of the edge to search for</td>
</tr>
<tr>
<td>nSearchLines</td>
<td>UDINT</td>
<td>Width of the search window, i.e. the number of search lines (1, 3, 5, 7, ..., centered around the line specified by aStartPoint and aEndPoint)</td>
</tr>
<tr>
<td>fSearchLineDistance</td>
<td>REAL</td>
<td>Distance between the search lines in pixels (&gt; 0)</td>
</tr>
<tr>
<td>nMaxThickness</td>
<td>UDINT</td>
<td>Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels</td>
</tr>
<tr>
<td>nSubpixelsIterations</td>
<td>UDINT</td>
<td>Specifies the number of subpixels (for INTERPOLATION, 10 - 20 usually is sufficient) or maximum number of iterations for optimizing the parameters (for APPROX_ERF and APPROX_GAUSSIAN, 50 - 100 usually is sufficient)</td>
</tr>
<tr>
<td>fApproxPrecision</td>
<td>REAL</td>
<td>Specifies the approximation precision for APPROX_ERF and APPROX_GAUSSIAN (0.001 usually is sufficient, unused for INTERPOLATION)</td>
</tr>
<tr>
<td>eAlgorithm</td>
<td>ETcVnEdgeDetectionAlgorithm [185]</td>
<td>Selection of the edge detection algorithm</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fAvgStrength</td>
<td>REAL</td>
<td>Returns the average strength of the detected edge</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Further information

The function F_VN_LocateEdgeExp is the expert version of F_VN_LocateEdge [1230]. It contains additional parameters.

### Parameter

**Input image**

The input image ipSrcImage must be a 1-channel grayscale image.
Localized points

(Return value)

The container ipEdgePoints returns the localized points of the linear edge.

The search window

First, a search window has to be defined to determine where and in which direction to search for an edge. This is done with the parameters aStartPoint, aEndPoint and nSearchLines. The search window results from the search line from aStartPoint to aEndPoint and parallel to it (nSearchLines - 1) / 2 search lines in each direction, each with 1 pixel distance. Thus nSearchLines defines the number of search lines and must always have an odd value.

Fig. 23: S: StartPoint, E: EndPoint, W: WindowWidth

Edge strength

(Return value)

The return value fAvgStrength displays the actual contrast of the localized edge (averaged over all search lines).

Parameters for edge localization

The remaining parameters are explained in detail in the chapter Edge localization [1216].

Application

The localization of a linear edge with expert parameters looks like this, for example:

```c
hr := F_VN_LocateEdgeExp( 
    ipSrcImage := ipImageIn, 
    ipEdgePoints := ipEdgePoints, 
    aStartPoint := aStartPoint, 
    aEndPoint := aEndPoint, 
    eEdgeDirection := eDirection, 
    fMinStrength := fMinStrength, 
    nSearchLines := nSearchLines, 
    fSearchLineDist := fSearchLineDist, 
    nMaxThickness := nMaxThickness, 
    nSubpixelsIterations := nSubpixIter, 
    fApproxPrecision := 0.0001, 
    eAlgorithm := eAlgorithm, 
    hrPrev := hr, 
    fAvgStrength := fAvgStrength 
);  
```

Samples

- Locate Edge [1427]
Related functions

- F_VN_LocateCircularArc(Exp) \[1227\] for localizing proportional circular arcs
- F_VN_LocateEdge(Exp) \[1233\] for localizing linear edges
- F_VN_LocateEdges(Exp) \[1240\] for localizing several linear edges
- F_VN_LocateEllipse(Exp) \[1244\] for localizing complete ellipses

Required License

TC3 Vision Metrology 2D

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.18.7 F_VN_LocateEdges

Locate the points of multiple occurring edges inside a specified search window, using an interpolation method.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

**Definition:**

FUNCTION F_VN_LocateEdges : HRESULT
VAR_INPUT
  ipSrcImage : ITCvImage;
  ipEdgePoints : Reference To ITcVnContainer;
  aStartPoint : Reference To TcVnPoint2_REAL;
  aEndPoint : Reference To TcVnPoint2_REAL;
  nNumEdges : UDINT;
  eEdgeDirection : ETcVnEdgeDirection;
  bAlternateDirection : BOOL;
  fMinStrength : REAL;
  nSearchLines : UDINT;
  hrPrev : HRESULT;
END_VAR

VAR_OUTPUT
  fAvgStrength : REAL;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage  [383]</td>
<td>Source image (1 channel)</td>
</tr>
<tr>
<td>ipEdgePoints</td>
<td>Reference To ITcVnContainer  [345]</td>
<td>Returns the detected edge points</td>
</tr>
<tr>
<td>aStartPoint</td>
<td>Reference To TcVnPoint2_REAL  [151]</td>
<td>Position from which to start the search process (in the direction of aEndPoint)</td>
</tr>
<tr>
<td>aEndPoint</td>
<td>Reference To TcVnPoint2_REAL  [151]</td>
<td>Position where the search process ends</td>
</tr>
<tr>
<td>nNumEdges</td>
<td>UDINT</td>
<td>The (maximum) number of edges to search for</td>
</tr>
<tr>
<td>eEdgeDirection</td>
<td>ETcVnEdgeDirection [186]</td>
<td>Specification of the edge direction to search for</td>
</tr>
<tr>
<td>bAlternateDirec tion</td>
<td>BOOL</td>
<td>If true, eEdgeDirection is alternated after each detected edge. Else, only edges with eEdgeDirection are searched for.</td>
</tr>
<tr>
<td>fMinStrength</td>
<td>REAL</td>
<td>Specification of the minimum strength (intensity difference) of the edge to search for</td>
</tr>
<tr>
<td>nSearchLines</td>
<td>UDINT</td>
<td>Width of the search window, i.e. the number of search lines (1, 3, 5, 7, ...) centered around the line specified by aStartPoint and aEndPoint</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fAvgStrength</td>
<td>REAL</td>
<td>Returns the average strength of the detected edges</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Further information

The function F_VN_LocateEdges localizes several edges in a defined search window.

### Parameter

#### Input image

The input image ipSrcImage must be a 1-channel grayscale image.

#### Localized points

The container ipEdgePoints returns the localized points of the linear edges.

#### The search window

First, a search window has to be defined to determine where and in which direction to search for an edge. This is done with the parameters aStartPoint, aEndPoint and nSearchLines. The search window results from the search line from aStartPoint to aEndPoint and parallel to it (nSearchLines - 1) / 2 search lines in each direction, each with 1 pixel distance. Thus nSearchLines defines the number of search lines and must always have an odd value.
Fig. 24: S: StartPoint, E: EndPoint, W: WindowWidth

**Edge strength** (Return value)

The return value \( f_{\text{AvgStrength}} \) displays the actual contrast of the localized edges (averaged over all search lines).

**Parameters for edge localization**

The remaining parameters are explained in detail in the chapter Edge localization [1216].

**Expert parameters**

The expert version F_VN_LocateEdgeExp contains additional parameters.

**Application**

The localization of a linear edge looks like this, for example:

```plaintext
hr := F_VN_LocateEdges(  
ipSrcImage := ipImageIn,  
ipEdgePoints := ipEdgePoints,  
aStartPoint := aStartPoint,  
aEndPoint := aEndPoint,  
nNumEdges := nNumEdges,  
eEdgeDirection := eDirection,  
bAlternateDirection := bAlternateDirection,  
fMinStrength := fMinStrength,  
nSearchLines := nSearchLines,  
hrPrev := hr,  
fAvgStrength => fAvgStrength);
```

The localized edge points can be visualized with the function F_VN_DrawContours [891].

```plaintext
F_VN_DrawContours(ipEdgePoints, -1, ipImageRes, aColorGreen, 5, hr);
```

**Related functions**

- F_VN_LocateCircularArc(Exp) [1227] for localizing proportional circular arcs
- F_VN_LocateEdge(Exp) [1233] for localizing linear edges
- F_VN_LocateEdges(Exp) [1240] for localizing several linear edges
- F_VN_LocateEllipse(Exp) [1244] for localizing complete ellipses
6.4.18.8 F_VN_LocateEdgesExp

**F_VN_LocateEdgesExp**

functions:

```
FUNCTION F_VN_LocateEdgesExp : HRESULT
VAR_INPUT
  ipSrcImage : ITCvImage;
  ipEdgePoints : Reference To ITCvContainer;
  aStartPoint : Reference To TcVnPoint2_REAL;
  aEndPoint : Reference To TcVnPoint2_REAL;
  nNumEdges : UDINT;
  eEdgeDirection : ETcVnEdgeDirection;
  bAlternateDirection : BOOL;
  fMinStrength : REAL;
  nSearchLines : UDINT;
  fSearchLineDist : REAL;
  nMaxThickness : UDINT;
  nSubpixelsIterations : UDINT;
  fApproxPrecision : REAL;
  eAlgorithm : ETcVnEdgeDetectionAlgorithm;
VAR_OUTPUT
  hrPrev : HRESULT;
  fAvgStrength : REAL;
END_VAR
```

Locate the points of multiple occurring edges inside a specified search window with subpixel accuracy.

(expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

**Syntax**

**Definition:**

```
FUNCTION F_VN_LocateEdgesExp : HRESULT
VAR_INPUT
  ipSrcImage : ITCvImage;
  ipEdgePoints : Reference To ITCvContainer;
  aStartPoint : Reference To TcVnPoint2_REAL;
  aEndPoint : Reference To TcVnPoint2_REAL;
  nNumEdges : UDINT;
  eEdgeDirection : ETcVnEdgeDirection;
  bAlternateDirection : BOOL;
  fMinStrength : REAL;
  nSearchLines : UDINT;
  fSearchLineDist : REAL;
  nMaxThickness : UDINT;
  nSubpixelsIterations : UDINT;
  fApproxPrecision : REAL;
  eAlgorithm : ETcVnEdgeDetectionAlgorithm;
VAR_OUTPUT
  hrPrev : HRESULT;
  fAvgStrength : REAL;
END_VAR
```

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage[383]</td>
<td>Source image (1 channel)</td>
</tr>
<tr>
<td>ipEdgePoints</td>
<td>Reference To ITcVnContainer[345]</td>
<td>Returns the detected edge points (ContainerType_Vector_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>aStartPoint</td>
<td>Reference To TcVnPoint2_REAL[151]</td>
<td>Position from which to start the search process (in the direction of aEndPoint)</td>
</tr>
<tr>
<td>aEndPoint</td>
<td>Reference To TcVnPoint2_REAL[151]</td>
<td>Position where the search process ends</td>
</tr>
<tr>
<td>nNumEdges</td>
<td>UDINT</td>
<td>The (maximum) number of edges to search for</td>
</tr>
<tr>
<td>eEdgeDirection</td>
<td>EITcVnEdgeDirection[186]</td>
<td>Specification of the edge direction to search for</td>
</tr>
<tr>
<td>bAlternateDirection</td>
<td>BOOL</td>
<td>If true, eEdgeDirection is alternated after each detected edge. Else, only edges with eEdgeDirection are searched for.</td>
</tr>
<tr>
<td>fMinStrength</td>
<td>REAL</td>
<td>Specification of the minimum strength (intensity difference) of the edge to search for</td>
</tr>
<tr>
<td>nSearchLines</td>
<td>UDINT</td>
<td>Width of the search window, i.e. the number of search lines (1, 3, 5, 7, ...), centered around the line specified by aStartPoint and aEndPoint</td>
</tr>
<tr>
<td>fSearchLineDistance</td>
<td>REAL</td>
<td>Distance between the search lines in pixels (&gt; 0)</td>
</tr>
<tr>
<td>nMaxThickness</td>
<td>UDINT</td>
<td>Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels</td>
</tr>
<tr>
<td>nSubpixelsIterations</td>
<td>UDINT</td>
<td>Specifies the number of subpixels (for INTERPOLATION, 10 - 20 usually is sufficient) or maximum number of iterations for optimizing the parameters (for APPROX_ERF and APPROX_GAUSSIAN, 50 - 100 usually is sufficient)</td>
</tr>
<tr>
<td>fApproxPrecision</td>
<td>REAL</td>
<td>Specifies the approximation precision for APPROX_ERF and APPROX_GAUSSIAN (0.001 usually is sufficient, unused for INTERPOLATION)</td>
</tr>
<tr>
<td>eAlgorithm</td>
<td>EITcVnEdgeDetectionAlgorithm[185]</td>
<td>Selection of the edge detection algorithm</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fAvgStrength</td>
<td>REAL</td>
<td>Returns the average strength of the detected edges</td>
</tr>
</tbody>
</table>

## Return value

HRESULT[135]

## Further information

The function F_VN LocateEdgesExp is the expert version of F_VN LocateEdges[1236]. It contains additional parameters.
Parameter

Input image

The input image `ipSrcImage` must be a 1-channel grayscale image.

Localized points (Return value)

The container `ipEdgePoints` returns the localized points of the linear edges.

The search window

First, a search window has to be defined to determine where and in which direction to search for an edge. This is done with the parameters `aStartPoint`, `aEndPoint` and `nSearchLines`. The search window results from the search line from `aStartPoint` to `aEndPoint` and parallel to it `(nSearchLines - 1) / 2` search lines in each direction, each with 1 pixel distance. Thus `nSearchLines` defines the number of search lines and must always have an odd value.

![Fig. 25: S: StartPoint, E: EndPoint, W: WindowWidth](image)

Edge strength (Return value)

The return value `fAvgStrength` displays the actual contrast of the localized edges (averaged over all search lines).

Parameters for edge localization

The remaining parameters are explained in detail in the chapter Edge localization [1216].

Application

The localization of a linear edge with expert parameters looks like this, for example:

```c
hr := F_VN_LocateEdgesExp(
  ipSrcImage := ipImageIn,
  ipEdgePoints := ipEdgePoints,
  aStartPoint := aStartPoint,
  aEndPoint := aEndPoint,
  nNumEdges := nNumEdges,
  eEdgeDirection := eDirection,
  bAlternateDirection := bAlternateDirection,
  fMinStrength := fMinStrength,
  nSearchLines := nSearchLines,
  fSearchLineDist := fSearchLineDist
  nMaxThickness := nMaxThickness,
  nSubpixelsIterations := nSubpixIter,
  fApproxPrecision := 0.001,
```
Related functions

- F_VN_LocateCircularArc(Exp) [1227] for localizing proportional circular arcs
- F_VN_LocateEdge(Exp) [1233] for localizing linear edges
- F_VN_LocateEdges(Exp) [1240] for localizing several linear edges
- F_VN_LocateEllipse(Exp) [1244] for localizing complete ellipses

Required License

TC3 Vision Metrology 2D

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g.</td>
<td>Tc3_Vision</td>
</tr>
<tr>
<td></td>
<td>Intel 4-core Atom CPU</td>
<td></td>
</tr>
</tbody>
</table>

6.4.18.9 F_VN_LocateEllipse

Locate an ellipse, using an interpolation method for locating the edges. Can use available TwinCAT Job Tasks for executing parallel code regions. Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_LocateEllipse : HRESULT
VAR_INPUT
  ipSrcImage : ITcVnImage;
  stEllipse : Reference To TcVnRotatedRectangle;
  aCenter : Reference To TcVnPoint2_REAL;
  fSearchRadius : REAL;
  eEdgeDirection : ETcVnEdgeDirection;
  fMinStrength : REAL;
  hrPrev : HRESULT;
END_VAR
```
The function `F_VN_LocateEllipse` localizes and measures an ellipse. It is based on the edge localization and uses normal parameters. It returns the localized ellipse as a structure of the type `TcVnRotatedRectangle` (center point, size and angle).

### Parameter

**Input image**

The input image `ipSrcImage` must be a 1-channel grayscale image.

**Ellipse (Return value)**

The parameter `stEllipse` returns the localized ellipse as a structure of the type `TcVnRotatedRectangle` (center point, size and angle).

**Search circle**

The circular search area is defined via the expected center point `aCenter` and a search radius `fSearchRadius`. The search radius should be set somewhat larger than the expected ellipse radius in order to securely localize the ellipse.

**Parameters for edge localization**

The remaining parameters are explained in detail in the chapter `Edge localization`.

**Expert parameters**

The expert version `F_VN_LocateEllipse(Exp)` contains additional parameters.

**Application**

The localization and measurement of an ellipse with an estimated center point of `[360, 240]` and a radius of 100 pixels looks like this, for example: for example:
VAR
  aGuessedCenter : TcVnPoint2_REAL := [360, 240];
  fGuessedRadius : REAL := 100;
END_VAR

hr := F_VN_LocateEllipseExp(
  ipSrcImage := ipImageIn,
  stEllipse := stEllipse,
  aCenterPoint := aGuessedCenter,
  fSearchRadius := (fGuessedRadius + 15), // increase search radius to make sure
ellipse is found!
  eEdgeDirection := TCVN_ED_LIGHT_TO_DARK,
  fMinStrength := fMinStrength,
  hrPrev := hr
);

The properties of the localized ellipse can then be retrieved via the variables stEllipse.aCenter, stEllipse.stSize and stEllipse.fAngle. In addition, the ellipse can be visualized with the function F_VN_DrawEllipse [894]:

hr := F_VN_DrawEllipse(stEllipse, ipImageRes, aColorGreen, 5, hr);

Samples
- Locate Ellipse [1432]

Related functions
- F_VN LocateCircularArc(Exp) [1227] for localizing proportional circular arcs
- F_VN LocateEdge(Exp) [1233] for localizing linear edges
- F_VN LocateEdges(Exp) [1234] for localizing several linear edges
- F_VN LocateEllipse(Exp) [1240] for localizing complete ellipses

Required License
TC3 Vision Metrology 2D

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.18.10 F_VN_LocateEllipseExp

```c
F_VN_LocateEllipseExp
```

- ipSrcImage : ITpImage
- stEllipse : Reference To TcVnRotatedRectangle
- aCenter : Reference To TcVnPoint2_REAL
- fSearchRadius : REAL
- eEdgeDirection : EtcVnEdgeDirection
- fMinStrength : REAL
- fMaxThickness : UDINT
- bInvertSearchDirection : BOOL
- fMinSearchRadius : REAL
- nSubpixelsIterations : UDINT
- nSearchLines : UDINT
- fApproxPrecision : REAL
- eAlgorithm : EtcVnEdgeDetectionAlgorithm
- ipContourPoints : Reference To ITCvNContainer
- hrPrev : HRESULT
Locate an ellipse. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

FUNCTION F_VN_LocateEllipseExp : HRESULT
VAR_INPUT
   ipSrcImage : ITcVnImage;
   stEllipse : Reference To TcVnRotatedRectangle;
   aCenter : Reference To TcVnPoint2_REAL;
   fSearchRadius : REAL;
   eEdgeDirection : ETcVnEdgeDirection;
   fMinStrength : REAL;
   nMaxThickness : UDINT;
   bInvertSearchDirection : BOOL;
   fMinSearchRadius : REAL;
   nSubpixelsIterations : UDINT;
   nSearchLines : UDINT;
   fApproxPrecision : REAL;
   eAlgorithm : ETcVnEdgeDetectionAlgorithm;
   ipContourPoints : Reference To ITcVnContainer;
   hrPrev : HRESULT;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td><a href="#">ITcVnImage</a></td>
<td>Source image (1 channel)</td>
</tr>
<tr>
<td>stEllipse</td>
<td>Reference To <a href="#">TcVnRotatedRectangle</a></td>
<td>Returns the detected ellipse</td>
</tr>
<tr>
<td>aCenter</td>
<td>Reference To <a href="#">TcVnPoint2_REAL</a></td>
<td>The expected ellipse center</td>
</tr>
<tr>
<td>fSearchRadius</td>
<td>REAL</td>
<td>Radius around aCenter to search for edges</td>
</tr>
<tr>
<td>eEdgeDirection</td>
<td><a href="#">ETcVnEdgeDirection</a></td>
<td>Specification of the edge direction to search for (from center to outside ellipse or other way round, if blnvertSearchDirection is true)</td>
</tr>
<tr>
<td>fMinStrength</td>
<td>REAL</td>
<td>Specification of the minimum strength (intensity difference) of the edge to search for</td>
</tr>
<tr>
<td>nMaxThickness</td>
<td>UDINT</td>
<td>Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels</td>
</tr>
<tr>
<td>blnvertSearchDirection</td>
<td>BOOL</td>
<td>If true, the search starts from outside the ellipse in direction of the center</td>
</tr>
<tr>
<td>fMinSearchRadius</td>
<td>REAL</td>
<td>Radius around aCenter to skip before starting to search for edges (e.g. to save time or if the center contains edges that should be ignored)</td>
</tr>
<tr>
<td>nSubpixelsIterations</td>
<td>UDINT</td>
<td>Specifies the number of subpixels (for INTERPOLATION, 10 - 20 usually is sufficient) or maximum number of iterations for optimizing the parameters (for APPROX_ERF and APPROX_GAUSSIAN, 50 - 100 usually is sufficient)</td>
</tr>
<tr>
<td>nSearchLines</td>
<td>UDINT</td>
<td>Specifies the amount of search lines, which are equally distributed in all directions (must be &gt;= 8 and a multiple of 4)</td>
</tr>
<tr>
<td>fApproxPrecision</td>
<td>REAL</td>
<td>Specifies the approximation precision for APPROX_ERF and APPROX_GAUSSIAN (0.001 usually is sufficient, unused for INTERPOLATION)</td>
</tr>
<tr>
<td>eAlgorithm</td>
<td><a href="#">ETcVnEdgeDetectionAlgorithm</a></td>
<td>Selection of the edge detection algorithm</td>
</tr>
<tr>
<td>ipContourPoints</td>
<td>Reference To <a href="#">ITcVnContainer</a></td>
<td>Returns the subpixel accurate ellipse contour (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2_REAL; Non-zero interface pointers are reused.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Further information

The function F_VN_LocateEllipseExp is the expert version of F_VN_LocateEllipse [1242].

### Parameter

### Input image

The input image ipSrcImage must be a 1-channel grayscale image.

### Ellipse

(Return value)
The parameter stEllipse returns the localized ellipse as structure of the type TcVnRotatedRectangle[222] (center point, size and angle).

**Search circle**

The circular search area is defined via the expected center point aCenter and a search radius fSearchRadius. The search radius should be set somewhat larger than the expected ellipse radius in order to securely localize the ellipse.

**Localized points** (Expert, return value)

The container ipContourPoints returns the localized points.

The parameter can be set to 0 if the return of the edge points is not required.

**Parameters for edge localization**

The remaining parameters are explained in detail in the chapter [Edge localization][1216].

**Application**

The localization and measurement of an ellipse with expert parameters looks like this, for example:

```csharp
hr := F_VN_LocateEllipseExp(
  ipSrcImage := ipImageIn,
  stEllipse := stEllipse,
  aCenterPoint := aGuessedCenter,
  fSearchRadius := (fGuessedRadius + 15), // increase search radius to make sure
  ellipse is found!
  eEdgeDirection := TCVN_ED_LIGHT_TO_DARK,
  fMinStrength := fMinStrength,
  nMaxThickness := 7,
  bInvertSearchDirection := TRUE,
  fMinSearchRadius := fMinRadius,
  nSubpixelsIterations := nSubpixIter,
  nSearchLines := nSearchLines,
  fApproxPrecision := 0.0001,
  eAlgorithm := eAlgorithm,
  ipContourPoints := ipContourPoints,
  hrPrev := hr
);
```

The localized points of the ellipse can be visualized with the function F_VN_DrawContours[890].

```csharp
hr := F_VN_DrawContours(ipContourPoints, -1, ipImageRes, aColorGreen, 5, hr);
```

**Samples**

- **Locate Ellipse [1432]**

**Related functions**

- F_VN_LocateCircularArc(Exp)[1227] for localizing proportional circular arcs
- F_VN_LocateEdge(Exp)[1233] for localizing linear edges
- F_VN_LocateEdges(Exp)[1240] for localizing several linear edges
- F_VN_LocateEllipse(Exp)[1244] for localizing complete ellipses

**Required License**

TC3 Vision Metrology 2D

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
Measure the angle between 2 edges, using an interpolation method.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

**Definition:**

```c
FUNCTION F_VN_MeasureAngleBetweenEdges : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    fAngle : Reference To REAL;
    aInnerPoint : Reference To TcVnPoint2_REAL;
    aOuterPoint1 : Reference To TcVnPoint2_REAL;
    aOuterPoint2 : Reference To TcVnPoint2_REAL;
    eEdgeDirection : ETcVnEdgeDirection;
    fMinStrength : REAL;
    nSearchLines : UDINT;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image (1 channel)</td>
</tr>
<tr>
<td>fAngle</td>
<td>Reference To REAL</td>
<td>Returns the angle between the detected edges (radians)</td>
</tr>
<tr>
<td>aInnerPoint</td>
<td>Reference To TcVnPoint2_REAL [151]</td>
<td>Position inside the angle to measure, from which to start the edge search process (in the direction of aOuterPoint1 and aOuterPoint2)</td>
</tr>
<tr>
<td>aOuterPoint1</td>
<td>Reference To TcVnPoint2_REAL [151]</td>
<td>Position where the search process for edge 1 ends</td>
</tr>
<tr>
<td>aOuterPoint2</td>
<td>Reference To TcVnPoint2_REAL [151]</td>
<td>Position where the search process for edge 2 ends</td>
</tr>
<tr>
<td>eEdgeDirection</td>
<td>ETcVnEdgeDirection [186]</td>
<td>Specification of the edge direction from aInnerPoint to aOuterPoint1 and aOuterPoint2 to search for</td>
</tr>
<tr>
<td>fMinStrength</td>
<td>REAL</td>
<td>Specification of the minimum strength (intensity difference) of the edge to search for</td>
</tr>
<tr>
<td>nSearchLines</td>
<td>UDINT</td>
<td>Width of the search windows, i.e. the number of search lines (3, 5, 7, ...), centered around the lines specified by aInnerPoint and aOuterPoint1, aOuterPoint2</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]
Further information

The function `F_VN_MeasureAngleBetweenEdges` localizes two linear edges and measures the angle between them.

Parameter

Input image

The input image `ipSrcImage` must be a 1-channel grayscale image.

Angle (Return value)

The parameter `fAngle` returns the angle between the localized edges.

Depending on the parameter `bAngleInDegrees`, the angle `fAngle` is specified in radians (FALSE) or in degrees (TRUE).

Parameters for edge localization

The remaining parameters are explained in detail in the chapter [Edge localization](1216).

Expert parameters

The expert version `F_VN_MeasureAngleBetweenEdgesExp` contains additional parameters.

Application

The measurement of an angle between two linear edges looks like this, for example:

```cpp
hr := F_VN_MeasureAngleBetweenEdgesExp(
   ipSrcImage := ipImageIn,
   fAngle := fAngle,
   aInnerPoint := aInnerPoint,
   aOuterPoint1 := aOuterPoint1,
   aOuterPoint2 := aOuterPoint2,
   eEdgeDirection := TCVN_ED_LIGHT_TO_DARK,
   fMinStrength := fMinStrength,
   nSearchLines := nSearchLines,
   hrPrev := hr
);
```

Samples

- Measure Angle Between Edges [1437]

Required License

TC3 Vision Metrology 2D

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
Measure the angle between 2 edges.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

FUNCTION F_VN_MeasureAngleBetweenEdgesExp : HRESULT
VAR_INPUT
   ipSrcImage          : ITCvnImage;
   fAngle              : Reference To REAL;
   aInnerPoint         : Reference To TcVnPoint2_REAL;
   aOuterPoint1        : Reference To TcVnPoint2_REAL;
   aOuterPoint2        : Reference To TcVnPoint2_REAL;
   eEdgeDirection      : ETcVnEdgeDetectionDirection;
   fMinStrength        : REAL;
   nSearchLines        : UDINT;
   fSearchLineDist     : REAL;
   nSubpixelsIterations : UDINT;
   bAngleInDegrees     : BOOL;
   fApproxPrecision    : REAL;
   eAlgorithm          : ETcVnEdgeDetectionAlgorithm;
   ipEdgePoints1       : Reference To ITCvnContainer;
   ipEdgePoints2       : Reference To ITCvnContainer;
   hrPrev              : HRESULT;
END_VAR
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage [383]</td>
<td>Source image (1 channel)</td>
</tr>
<tr>
<td>fAngle</td>
<td>Reference To REAL</td>
<td>Returns the angle between the detected edges (radians or degrees, depending on bAngleInDegrees)</td>
</tr>
<tr>
<td>aInnerPoint</td>
<td>Reference To TcVnPoint2_REAL [151]</td>
<td>Position inside the angle to measure, from which to start the edge search process (in the direction of aOuterPoint1 and aOuterPoint2)</td>
</tr>
<tr>
<td>aOuterPoint1</td>
<td>Reference To TcVnPoint2_REAL [151]</td>
<td>Position where the search process for edge 1 ends</td>
</tr>
<tr>
<td>aOuterPoint2</td>
<td>Reference To TcVnPoint2_REAL [151]</td>
<td>Position where the search process for edge 2 ends</td>
</tr>
<tr>
<td>eEdgeDirection</td>
<td>ETcVnEdgeDirection [186]</td>
<td>Specification of the edge direction from aInnerPoint to aOuterPoint1 and aOuterPoint2 to search for</td>
</tr>
<tr>
<td>fMinStrength</td>
<td>REAL</td>
<td>Specification of the minimum strength (intensity difference) of the edge to search for</td>
</tr>
<tr>
<td>nSearchLines</td>
<td>UDINT</td>
<td>Width of the search window, i.e. the number of search lines (3, 5, 7, ...), centered around the line specified by aInnerPoint and aOuterPoint1, aOuterPoint2</td>
</tr>
<tr>
<td>fSearchLineDistance</td>
<td>REAL</td>
<td>Distance between the search lines in pixels (&gt; 0)</td>
</tr>
<tr>
<td>nMaxThickness</td>
<td>UDINT</td>
<td>Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels</td>
</tr>
<tr>
<td>bInvertSearch</td>
<td>BOOL</td>
<td>If true, the search starts from each aOuterPoint in direction of aInnerPoint</td>
</tr>
<tr>
<td>nSubpixelsIterations</td>
<td>UDINT</td>
<td>Specifies the number of subpixels (for INTERPOLATION, 10 - 20 usually is sufficient) or maximum number of iterations for optimizing the parameters (for APPROX_ERF and APPROX_GAUSSIAN, 50 - 100 usually is sufficient)</td>
</tr>
<tr>
<td>bAngleInDegrees</td>
<td>BOOL</td>
<td>fAngle is in degrees, if true</td>
</tr>
<tr>
<td>fApproxPrecision</td>
<td>REAL</td>
<td>Specifies the approximation precision for APPROX_ERF and APPROX_GAUSSIAN (0.001 usually is sufficient, unused for INTERPOLATION)</td>
</tr>
<tr>
<td>eAlgorithm</td>
<td>ETcVnEdgeDetectionAlgorithm [185]</td>
<td>Selection of the edge detection algorithm</td>
</tr>
<tr>
<td>ipEdgePoints1</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns the detected edge points between aInnerPoint and aOuterPoint1 (ContainerType_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>ipEdgePoints2</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns the detected edge points between aInnerPoint and aOuterPoint2 (ContainerType_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Further information

The function `F_VN_MeasureAngleBetweenEdgesExp` is the expert function of `F_VN_MeasureAngleBetweenEdges` [1248].
Parameter

Input image

The input image `ipSrcImage` must be a 1-channel grayscale image.

Angle (Return value)

The parameter `fAngle` returns the angle between the localized edges.

Depending on the parameter `bAngleInDegrees`, the angle `fAngle` is specified in radians (FALSE) or in degrees (TRUE).

Parameters for edge localization

The remaining parameters are explained in detail in the chapter Edge localization [1216].

Localized points (Expert, return value)

The containers `ipEdgePoints1` and `ipEdgePoints2` return the exact points that were localized on the basis of each search line. If not needed, the value 0 can be transferred instead of an interface pointer.

Parameters for edge localization

The remaining parameters are explained in detail in the chapter Edge localization [1216].

Application

The measurement of an angle with expert parameters looks like this, for example:

```c
hr := F_VN_MeasureAngleBetweenEdgesExp(
    ipSrcImage := ipImageIn,
    fAngle := fAngle,
    aInnerPoint := aInnerPoint,
    aOuterPoint1 := aOuterPoint1,
    aOuterPoint2 := aOuterPoint2,
    eEdgeDirection := TCVN_ED_LIGHT_TO_DARK,
    fMinStrength := fMinStrength,
    nSearchLines := nSearchLines,
    fSearchLineDist := fSearchLineDist,
    nMaxThickness := 7,
    nSubpixelsIterations := 100,
    bAngleInDegrees := TRUE,
    fApproxPrecision := 0.0001,
    eAlgorithm := TCVN_EDA_APPROX_ERF,
    ipEdgePoints1 := ipEdgePoints1,
    ipEdgePoints2 := ipEdgePoints2,
    hrPrev := hr);
```

Samples

- Measure Angle Between Edges [1437]

Required License

TC3 Vision Metrology 2D

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.18.13 F_VN_MeasureEdgeDistance

Measure the distance between 2 parallel edges, using an interpolation method. Can use available TwinCAT Job Tasks for executing parallel code regions. Can return partial results when canceled by Watchdog.

Syntax

Definition:
FUNCTION F_VN_MeasureEdgeDistance : HRESULT
VAR_INPUT
  ipSrcImage : ITCvNImage;
  fAvgDistance : Reference To REAL;
  aStartPoint : Reference To TcVnPoint2_REAL;
  aEndPoint : Reference To TcVnPoint2_REAL;
  eEdgeDirection : ETcVnEdgeDirection;
  fMinStrength : REAL;
  nSearchLines : UDINT;
  hrPrev : HRESULT;
END_VAR

Return value
HRESULT [135]

Further information
The function F_VN_MeasureEdgeDistance localizes two linear edges and measures the average distance between them.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvNImage [383]</td>
<td>Source image (1 channel)</td>
</tr>
<tr>
<td>fAvgDistance</td>
<td>Reference To REAL</td>
<td>Returns the average distance between the detected edges</td>
</tr>
<tr>
<td>aStartPoint</td>
<td>Reference To TcVnPoint2_REAL [151]</td>
<td>Position from which to start the search process (in the direction of aEndPoint)</td>
</tr>
<tr>
<td>aEndPoint</td>
<td>Reference To TcVnPoint2_REAL [151]</td>
<td>Position where the search process ends</td>
</tr>
<tr>
<td>eEdgeDirection</td>
<td>ETcVnEdgeDirection [186]</td>
<td>Specification of the edge direction from aStartPoint to aEndPoint to search for</td>
</tr>
<tr>
<td>fMinStrength</td>
<td>REAL</td>
<td>Specification of the minimum strength (intensity difference) of the edge to search for</td>
</tr>
<tr>
<td>nSearchLines</td>
<td>UDINT</td>
<td>Width of the search window, i.e. the number of search lines (1, 3, 5, 7, ...), centered around the line specified by aStartPoint and aEndPoint</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Parameter

Input image

The input image `ipSrcImage` must be a 1-channel grayscale image.

Minimum distance (Return value)

The parameter `fAvgDistance` returns the minimum distance between the two localized edges.

Parameters for edge localization

The remaining parameters are explained in detail in the chapter `Edge localization [1216].`

Expert parameters

The expert version `F_VN_MeasureEdgeDistanceExp [1255]` contains additional parameters.

Application

The measurement of an edge distance looks like this, for example:

```c
hr := F_VN_MeasureEdgeDistance(
    ipSrcImage := ipImageIn,
    fAvgDistance := fDistance,
    aStartPoint := aStartPoint,
    aEndPoint := aEndPoint,
    eEdgeDirection := TCVN_ED_LIGHT_TO_DARK,
    fMinStrength := fMinStrength,
    nSearchLines := nSearchLines,
    hrPrev := hr
);
```

Related functions

- `F_VN_ClosestPointsBF [1222]` for calculating the closest points in two point clouds
- `F_VN_MeasureAngleBetweenEdges(Exp) [1250]` for measuring the angle between two edges
- `F_VN_MeasureEdgeDistance(Exp) [1255]` for measuring the average distance between two edges
- `F_VN_MeasureMinEdgeDistance(Exp) [1260]` for measuring the minimum distance between two edges

Required License

TC3 Vision Metrology 2D

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
Measure the distance between 2 parallel edges. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

**Definition:**

```plaintext
FUNCTION F_VN_MeasureEdgeDistanceExp : HRESULT
VAR_INPUT
  ipSrcImage : ITCvImage;
  fAvgDistance : REAL;
  aStartPoint : Reference To TcVnPoint2_REAL;
  aEndPoint : Reference To TcVnPoint2_REAL;
  eEdgeDirection : ETcVnEdgeDirection;
  fMinStrength : REAL;
  nSearchLines : UDINT;
  fSearchLineDist : REAL;
  nMaxThickness : UDINT;
  bInvertSearchDirection : BOOL;
  fSearchGap : REAL;
  nSubpixelsIterations : UDINT;
  fApproxPrecision : REAL;
  eAlgorithm : ETcVnEdgeDetectionAlgorithm;
  ipEdgePoints1 : Reference To ITcVnContainer;
  ipEdgePoints2 : Reference To ITcVnContainer;
  ipDistances : Reference To ITcVnContainer;
  hrPrev : HRESULT;
END_VAR
```
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td><a href="#">ITcVnImage</a></td>
<td>Source image (1 channel)</td>
</tr>
<tr>
<td>fAvgDistance</td>
<td>Reference To REAL</td>
<td>Returns the average distance between the detected edges</td>
</tr>
<tr>
<td>aStartPoint</td>
<td>Reference To <a href="#">TcVnPoint2_REAL</a></td>
<td>Position from which to start the search process (in the direction of aEndPoint)</td>
</tr>
<tr>
<td>aEndPoint</td>
<td>Reference To <a href="#">TcVnPoint2-REAL</a></td>
<td>Position where the search process ends</td>
</tr>
<tr>
<td>eEdgeDirection</td>
<td><a href="#">ETcVnEdgeDirection</a></td>
<td>Specification of the edge direction from aStartPoint to aEndPoint to search for</td>
</tr>
<tr>
<td>fMinStrength</td>
<td>REAL</td>
<td>Specification of the minimum strength (intensity difference) of the edge to search for</td>
</tr>
<tr>
<td>nSearchLines</td>
<td>UDINT</td>
<td>Width of the search window, i.e. the number of search lines (1, 3, 5, 7, ...), centered around the line specified by aStartPoint and aEndPoint</td>
</tr>
<tr>
<td>fSearchLineDist</td>
<td>REAL</td>
<td>Distance between the search lines in pixels (&gt; 0)</td>
</tr>
<tr>
<td>nMaxThickness</td>
<td>UDINT</td>
<td>Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels</td>
</tr>
<tr>
<td>bInvertSearch</td>
<td>BOOL</td>
<td>If true, the search starts from the center point between aStartPoint and aEndPoint in both directions</td>
</tr>
<tr>
<td>fSearchGap</td>
<td>REAL</td>
<td>Optional width of a gap (&gt;= 0, centered between aStartPoint and aEndPoint), that is neglected for searching edges (can reduce execution time). The 2 edges to search for must be on different sides of the gap.</td>
</tr>
<tr>
<td>nSubpixelsIterations</td>
<td>UDINT</td>
<td>Specifies the number of subpixels (for INTERPOLATION, 10 - 20 usually is sufficient) or maximum number of iterations for optimizing the parameters (for APPROX_ERF and APPROX_GAUSSIAN, 50 - 100 usually is sufficient)</td>
</tr>
<tr>
<td>fApproxPrecision</td>
<td>REAL</td>
<td>Specifies the approximation precision for APPROX_ERF and APPROX_GAUSSIAN (0.001 usually is sufficient, unused for INTERPOLATION)</td>
</tr>
<tr>
<td>eAlgorithm</td>
<td><a href="#">ETcVnEdgeDetectionAlgorithm</a></td>
<td>Selection of the edge detection algorithm</td>
</tr>
<tr>
<td>ipEdgePoints1</td>
<td>Reference To <a href="#">ITcVnContainer</a></td>
<td>Returns the detected edge points of the edge near aStartPoint (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2-REAL)</td>
</tr>
<tr>
<td>ipEdgePoints2</td>
<td>Reference To <a href="#">ITcVnContainer</a></td>
<td>Returns the detected edge points of the edge near aEndPoint (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2-REAL)</td>
</tr>
<tr>
<td>ipDistances</td>
<td>Reference To <a href="#">ITcVnContainer</a></td>
<td>Returns the distances between the detected edge points (optional, set to 0 if not required; ContainerType_Vector-REAL)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [#]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [#]
Further information

The function \texttt{F\_VN\_MeasureEdgeDistanceExp} is the expert version of \texttt{F\_VN\_MeasureEdgeDistance}. It contains additional parameters.

Parameter

Input image

The input image \texttt{ipSrcImage} must be a 1-channel grayscale image.

Minimum distance (Return value)

The parameter \texttt{fAvgDistance} returns the minimum distance between the two localized edges.

Localized edge points (Expert, return value)

The containers \texttt{ipEdgePoints1} and \texttt{ipEdgePoints2} return the exact points that were localized on the basis of each search line. If not needed, the value 0 can be transferred instead of an interface pointer.

Measured distances (Expert, return value)

The container \texttt{ipDistances} returns all measured distances between opposite points. If not needed, the value 0 can be transferred instead of an interface pointer.

Search gap (Expert)

Area in the center of the defined search line in which the search is omitted. The execution time can be reduced as a result.

Parameters for edge localization

The remaining parameters are explained in detail in the chapter \texttt{Edge localization}.

Application

The measurement of an edge distance with expert parameters looks like this, for example:

\begin{verbatim}
hr := F\_VN\_MeasureEdgeDistance(
  ipSrcImage := ipImageIn,
  fAvgDistance := fDistance,
  aStartPoint := aStartPoint,
  aEndPoint := aEndPoint,
  eEdgeDirection := TCVN\_ED\_LIGHT\_TO\_DARK,
  fMinStrength := fMinStrength,
  nSearchLines := nSearchLines,
  fSearchLineDist := 3,
  nMaxThickness := 7,
  bInvertSearchDirection := TRUE,
  nSubpixelsIterations := nSubpixIter,
  fApproxPrecision := 0.0001,
  eAlgorithm := eAlgorithm,
  ipEdgePoints1 := ipEdge1,
  ipEdgePoints2 := ipEdge2,
  ipDistances := ipDistances,
  hrPrev := hr);
\end{verbatim}

The localized edge points can be visualized with the function \texttt{F\_VN\_DrawContours}.

\begin{verbatim}
hr := F\_VN\_DrawContours(ipEdgePoints1, -1, ipImageRes, aColorGreen, 5, hr);
hr := F\_VN\_DrawContours(ipEdgePoints2, -1, ipImageRes, aColorGreen, 5, hr);
\end{verbatim}

Related functions

- \texttt{F\_VN\_ClosestPointsBF} for calculating the closest points in two point clouds
- \texttt{F\_VN\_MeasureAngleBetweenEdges(Exp)} for measuring the angle between two edges
- \texttt{F\_VN\_MeasureEdgeDistance(Exp)} for measuring the average distance between two edges
- \texttt{F\_VN\_MeasureMinEdgeDistance(Exp)} for measuring the minimum distance between two edges
Required License

TC3 Vision Metrology 2D

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.18.15  F_VN_MeasureMinEdgeDistance

Measure the minimum distance within the specified search window between 2 edges, using an interpolation method.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

FUNCTION F_VN_MeasureMinEdgeDistance : HRESULT
VAR_INPUT
  ipSrcImage : ITCvImage;
  fMinDistance : Reference To REAL;
  aStartPoint : Reference To TcVnPoint2_REAL;
  aEndPoint : Reference To TcVnPoint2_REAL;
  eEdgeDirection : ETcVnEdgeDirection;
  fMinStrength : REAL;
  nSearchLines : UDINT;
  hrPrev : HRESULT;
END_VAR
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITcVnImage</td>
<td>Source image (1 channel)</td>
</tr>
<tr>
<td>fMinDistance</td>
<td>Reference To REAL</td>
<td>Returns the minimum distance between the detected edges</td>
</tr>
<tr>
<td>aStartPoint</td>
<td>Reference To TcVnPoint2_REAL</td>
<td>Position from which to start the search process (in the direction of aEndPoint)</td>
</tr>
<tr>
<td>aEndPoint</td>
<td>Reference To TcVnPoint2_REAL</td>
<td>Position where the search process ends</td>
</tr>
<tr>
<td>eEdgeDirection</td>
<td>ETcVnEdgeDirection</td>
<td>Specification of the edge direction from aStartPoint to aEndPoint to search for</td>
</tr>
<tr>
<td>fMinStrength</td>
<td>REAL</td>
<td>Specification of the minimum strength (intensity difference) of the edge to search for</td>
</tr>
<tr>
<td>nSearchLines</td>
<td>UDINT</td>
<td>Width of the search window, i.e. the number of search lines (1, 3, 5, 7, ...), centered around the line specified by aStartPoint and aEndPoint</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Further information

The function F_VN_MeasureMinEdgeDistance localizes two linear edges and measures the minimum distance between them. It returns the minimum distance as REAL.

### Parameter

**Input image**

The input image ipSrcImage must be a 1-channel grayscale image.

**Minimum distance (Return value)**

The parameter fMinDistance returns the minimum distance between the two localized edges.

### Parameters for edge localization

The remaining parameters are explained in detail in the chapter Edge localization [1216].

### Expert parameters

The expert version F_VN_MeasureMinEdgeDistanceExp [1260] contains additional parameters.

### Application

The measurement of an edge distance looks like this, for example:

```c
hr := F_VN_MeasureMinEdgeDistance(
    ipSrcImage := ipImageIn,
    fMinDistance := fDistance,
    aStartPoint := aStartPoint,
    aEndPoint := aEndPoint,
    eEdgeDirection := TCVN_ED_LIGHT_TO_DARK,
    fMinStrength := fMinStrength,
    nSearchLines := nSearchLines,
    hrPrev := hr
);
```
**Related functions**

- **F_VN_ClosestPointsBF** ([1222](#)) for calculating the closest points in two point clouds
- **F_VN_MeasureAngleBetweenEdges(Exp)** ([1250](#)) for measuring the angle between two edges
- **F_VN_MeasureEdgeDistance(Exp)** ([1255](#)) for measuring the average distance between two edges
- **F_VN_MeasureMinEdgeDistance(Exp)** ([1260](#)) for measuring the minimum distance between two edges

**Required License**

TC3 Vision Metrology 2D

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.18.16  F_VN_MeasureMinEdgeDistanceExp

**FUNCTION F_VN_MeasureMinEdgeDistanceExp : HRESULT**

```pascal
VAR_INPUT
  ipSrcImage : ITcVnImage;
  fMinDistance : Reference To REAL;
  aStartPoint : Reference To TcVnPoint2_REAL;
  aEndPoint : Reference To TcVnPoint2_REAL;
  eEdgeDirection : ETcVnEdgeDirection;
  fMinStrength : REAL;
  nSearchLines : UDINT;
  fSearchLineDist : REAL;
  nMaxThickness : UDINT;
  bInvertSearchDirection : BOOL;
  fSearchGap : REAL;
  nSubpixelsIterations : UDINT;
  fApproxPrecision : REAL;
  eAlgorithm : ETcVnEdgeDetectionAlgorithm;
  ipEdgePoints1 : Reference To ITcVnContainer;
  ipEdgePoints2 : Reference To ITcVnContainer;
  aPoint1 : Reference To TcVnPoint2_REAL;
  aPoint2 : Reference To TcVnPoint2_REAL;
  hrPrev : HRESULT;
```

Measure the minimum distance within the specified search window between 2 edges. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

**Syntax**

**Definition:**

```pascal
FUNCTION F_VN_MeasureMinEdgeDistanceExp : HRESULT
```

**Version:** 1.3
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrcImage</td>
<td>ITCvNImage[383]</td>
<td>Source image (1 channel)</td>
</tr>
<tr>
<td>fMinDistance</td>
<td>Reference To REAL</td>
<td>Returns the minimum distance between the detected edges</td>
</tr>
<tr>
<td>aStartPoint</td>
<td>Reference To TcVnPoint2_REAL[151]</td>
<td>Position from which to start the search process (in the direction of aEndPoint)</td>
</tr>
<tr>
<td>aEndPoint</td>
<td>Reference To TcVnPoint2_REAL[151]</td>
<td>Position where the search process ends</td>
</tr>
<tr>
<td>aEdgeDirection</td>
<td>ETcVnEdgeDirection[186]</td>
<td>Specification of the edge direction from aStartPoint to aEndPoint to search for</td>
</tr>
<tr>
<td>fMinStrength</td>
<td>REAL</td>
<td>Specification of the minimum strength (intensity difference) of the edge to search for</td>
</tr>
<tr>
<td>nSearchLines</td>
<td>UDINT</td>
<td>Width of the search window, i.e. the number of search lines (1, 3, 5, 7, ...), centered around the line specified by aStartPoint and aEndPoint</td>
</tr>
<tr>
<td>fSearchLineDist</td>
<td>REAL</td>
<td>Distance between the search lines in pixels (&gt; 0)</td>
</tr>
<tr>
<td>nMaxThickness</td>
<td>UDINT</td>
<td>Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels</td>
</tr>
<tr>
<td>bInvertSearch</td>
<td>BOOL</td>
<td>If true, the search starts from the center point between aStartPoint and aEndPoint in both directions</td>
</tr>
<tr>
<td>fSearchGap</td>
<td>REAL</td>
<td>Optional width of a gap (&gt;= 0, centered between aStartPoint and aEndPoint), that is neglected for searching edges (can reduce execution time). The 2 edges to search for must be on different sides of the gap.</td>
</tr>
<tr>
<td>nSubpixelsItera</td>
<td>UDINT</td>
<td>Specifies the number of subpixels (for INTERPOLATION, 10 - 20 usually is sufficient) or maximum number of iterations for optimizing the parameters (for APPROX_ERF and APPROX_GAUSSIAN, 50 - 100 usually is sufficient)</td>
</tr>
<tr>
<td>fApproxPrecison</td>
<td>REAL</td>
<td>Specifies the approximation precision for APPROX_ERF and APPROX_GAUSSIAN (0.001 usually is sufficient, unused for INTERPOLATION)</td>
</tr>
<tr>
<td>eAlgorithm</td>
<td>ETcVnEdgeDetectionAlgorithm[185]</td>
<td>Selection of the edge detection algorithm</td>
</tr>
<tr>
<td>ipEdgePoints1</td>
<td>Reference To ITCvNContainer[345]</td>
<td>Returns the detected edge points of the edge near aStartPoint (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>ipEdgePoints2</td>
<td>Reference To ITCvNContainer[345]</td>
<td>Returns the detected edge points of the edge near aEndPoint (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2_REAL)</td>
</tr>
<tr>
<td>aPoint1</td>
<td>Reference To TcVnPoint2_REAL[151]</td>
<td>Returns the point on the edge near aStartPoint, for which the minimum distance is achieved</td>
</tr>
<tr>
<td>aPoint2</td>
<td>Reference To TcVnPoint2_REAL[151]</td>
<td>Returns the point on the edge near aEndPoint, for which the minimum distance is achieved</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT[135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Further information

The function F_VN_MeasureMinEdgeDistanceExp is the expert version of F_VN_MeasureMinEdgeDistance [1258]. It contains additional parameters.

Parameter

Input image

The input image ipSrcImage must be a 1-channel grayscale image.

Minimum distance (Return value)

The parameter fMinDistance returns the minimum distance between the two localized edges.

Localized edge points (Return value)

The containers ipEdgePoints1 and ipEdgePoints2 return the exact points that were localized on the basis of each search line. If not needed, the value 0 can be transferred instead of an interface pointer.

In addition, the points aPoint1 and aPoint2 return the points that have the minimum distance between the two edges.

Search gap (Expert)

Area in the center of the defined search line in which the search is omitted. The execution time can be reduced as a result.

Parameters for edge localization

The remaining parameters are explained in detail in the chapter Edge localization [1216].

Application

The measurement of the minimum edge distance with expert parameters looks like this, for example:

```c
hr := F_VN_MeasureEdgeDistance(
    ipSrcImage := ipImageIn,
    fMinDistance := fDistance,
    aStartPoint := aStartPoint,
    aEndPoint := aEndPoint,
    eEdgeDirection := TCVN_ED_LIGHT_TO_DARK,
    fMinStrength := fMinStrength,
    nSearchLines := nSearchLines,
    fSearchLineDist := 3,
    nMaxThickness := 7,
    bInvertSearchDirection := TRUE,
    nSubpixelsIterations := 100,
    fApproxPrecision := 0.0001,
    eAlgorithm := TCVN_EDA_APPROX_ERF,
    ipEdgePoints1 := ipEdge1,
    ipEdgePoints2 := ipEdge2,
    aPoint1 := aPoint1,
    aPoint2 := aPoint2,
    hrPrev := hr);
```

The localized edge points can be visualized with the functions F_VN_DrawContours [890] and F_VN_DrawPoint [909]:

```c
hr := F_VN_DrawContours(ipEdgePoints1, -1, ipImageRes, aColorGreen, 5, hr);
hr := F_VN_DrawContours(ipEdgePoints2, -1, ipImageRes, aColorGreen, 5, hr);
F_VN_DrawPoint(aPoint1[0], aPoint1[1], ipImageRes, TCVN_DS_X, aColorGreen, hr);
F_VN_DrawPoint(aPoint2[0], aPoint2[1], ipImageRes, TCVN_DS_X, aColorGreen, hr);
```
Related functions

- **F_VN_ClosestPointsBF** [1222] for calculating the closest points in two point clouds
- **F_VN_MeasureAngleBetweenEdges(Exp)** [1250] for measuring the angle between two edges
- **F_VN_MeasureEdgeDistance(Exp)** [1255] for measuring the average distance between two edges
- **F_VN_MeasureMinEdgeDistance(Exp)** [1260] for measuring the minimum distance between two edges

Required License

TC3 Vision Metrology 2D

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.19 Miscellaneous

This group contains further functions that can be used across groups.

Functions

Geometrical auxiliary functions

- **F_VN_ClipLineToBoundary** [1264]
- **F_VN_ClipLineToBoundary_ITcVnImage** [1265]
- **F_VN_ClipLineToBoundary_TcVnRectangle_DINT** [1266]
- **F_VN_LineIntersectionPoint** [1275]
- **F_VN_LineIntersectionPointAndAngle** [1275]
- **F_VN_RotatedRectangleCorners** [1277]
- **F_VN_RotatedRectangleIntersection** [1278]

Timestamp

- **F_VN_GetTimestamp** [1271]
- **F_VN_UpdateTimestamp** [1280]

Auxiliary functions for matrices

- **F_VN_InitMatrixStruct** [1272]
- **F_VN_InvertMatrix3x3** [1274]
- **F_VN_MultiplyMatrices** [1276]

Converting interfaces

- **F_VN_ConvertITcUnknownToITcVnBitmapExport** [1267]
- **F_VN_ConvertITcUnknownToITcVnContainer** [1268]
- **F_VN_ConvertITcUnknownToITcVnImage** [1269]
- **F_VN_ConvertITcUnknownToITcVnMlModel** [1270]

Miscellaneous

- **F_VN_HuMomentInvariants** [1271]
- **F_VN_SetRngSeed** [1279]
Clips a line to a rectangular boundary.

**Syntax**

**Definition:**

```
FUNCTION F_VN_ClipLineToBoundary : HRESULT
VAR_INPUT
    aLine : Reference To TcVnVector4_LREAL;
    nX1  : DINT;
    nY1  : DINT;
    nX2  : DINT;
    nY2  : DINT;
    aStartPoint : Reference To TcVnPoint2_REAL;
    aEndPoint : Reference To TcVnPoint2_REAL;
    hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aLine</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>The first and second element describe the x and y component of the direction vector. The third and fourth element describe the x and y component of the position vector.</td>
</tr>
<tr>
<td>nX1</td>
<td>DINT</td>
<td>x of top left rectangle point</td>
</tr>
<tr>
<td>nY1</td>
<td>DINT</td>
<td>y of top left rectangle point</td>
</tr>
<tr>
<td>nX2</td>
<td>DINT</td>
<td>x of bottom right rectangle point</td>
</tr>
<tr>
<td>nY2</td>
<td>DINT</td>
<td>Y of bottom right rectangle point</td>
</tr>
<tr>
<td>aStartPoint</td>
<td>Reference To TcVnPoint2_REAL</td>
<td>Returns the clipped starting point of the line.</td>
</tr>
<tr>
<td>aEndPoint</td>
<td>Reference To TcVnPoint2_REAL</td>
<td>Returns the clipped end point of the line.</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

RESULT [135]

**Further information**

This function adjusts a line to a specified area. Different functions exist for various definition forms in this area:
Definition of the rectangular area | Function to be used
--- | ---
Specified by individual X/Y coordinates | F_VN_ClipLineToBoundary
By the size of an image | F_VN_ClipLineToBoundary_ITcVnImage
By a rectangular structure | F_VN_ClipLineToBoundary_TcVnRectangle_DINT

**Required License**
TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.19.2 F_VN_ClipLineToBoundary_ITcVnImage

Clips a line to the boundary of an image.

**Syntax**

Definition:

```plaintext
FUNCTION F_VN_ClipLineToBoundary_ITcVnImage : HRESULT
VAR_INPUT
  aLine : Reference To TcVnVector4_LREAL;
  ipImage : ITcVnImage;
  aStartPoint : Reference To TcVnPoint2_REAL;
  aEndPoint : Reference To TcVnPoint2_REAL;
  hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aLine</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>The first and second element describe the x and y component of the direction vector. The third and fourth element describe the x and y component of the position vector.</td>
</tr>
<tr>
<td>ipImage</td>
<td>ITcVnImage</td>
<td>Image, from which to derive the boundary</td>
</tr>
<tr>
<td>aStartPoint</td>
<td>Reference To TcVnPoint2_REAL</td>
<td>Returns the clipped starting point of the line.</td>
</tr>
<tr>
<td>aEndPoint</td>
<td>Reference To TcVnPoint2_REAL</td>
<td>Returns the clipped end point of the line.</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT
Further information

This function adjusts a line to a specified area.

Different functions exist for various definition forms in this area:

<table>
<thead>
<tr>
<th>Definition of the rectangular area</th>
<th>Function to be used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specified by individual X/Y coordinates</td>
<td>F_VN_ClipLineToBoundary</td>
</tr>
<tr>
<td>By the size of an image</td>
<td>F_VN_ClipLineToBoundary_ITcVnImage</td>
</tr>
<tr>
<td>By a rectangular structure</td>
<td>F_VN_ClipLineToBoundary_TcVnRectangle_DINT</td>
</tr>
</tbody>
</table>

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.19.3  F_VN_ClipLineToBoundary_TcVnRectangle_DINT

Clips a line to a rectangular boundary.

Syntax

Definition:

```plaintext
FUNCTION F_VN_ClipLineToBoundary_TcVnRectangle_DINT : HRESULT
VAR_INPUT
    aLine       : Reference To TcVnVector4_LREAL;
    stRectangle : Reference To TcVnRectangle_DINT;
    aStartPoint : Reference To TcVnPoint2_REAL;
    aEndPoint   : Reference To TcVnPoint2_REAL;
    hrPrev      : HRESULT;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aLine</td>
<td>Reference To TcVnVector4 LREAL</td>
<td>The first and second element describe the x and y component of the direction vector. The third and fourth element describe the x and y component of the position vector.</td>
</tr>
<tr>
<td>stRectangle</td>
<td>Reference To TcVnRectangle_DINT</td>
<td>Rectangular boundary</td>
</tr>
<tr>
<td>aStartPoint</td>
<td>Reference To TcVnPoint2_REAL</td>
<td>Returns the clipped starting point of the line.</td>
</tr>
<tr>
<td>aEndPoint</td>
<td>Reference To TcVnPoint2_REAL</td>
<td>Returns the clipped end point of the line.</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Further information**

This function adjusts a line to a specified area.

Different functions exist for various definition forms in this area:

<table>
<thead>
<tr>
<th>Definition of the rectangular area</th>
<th>Function to be used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specified by individual X/Y coordinates</td>
<td>F_VN_ClipLineToBoundary [1264]</td>
</tr>
<tr>
<td>By the size of an image</td>
<td>F_VN_ClipLineToBoundary ITcVnImage [1265]</td>
</tr>
<tr>
<td>By a rectangular structure</td>
<td>F_VN_ClipLineToBoundary TcVnRectangle_DINT [1266]</td>
</tr>
</tbody>
</table>

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.4.19.4 F_VN_ConvertITcUnknownToITcVnBitmapExport**

Convert an ITcUnknown interface pointer to an ITcVnBitmapExport interface pointer. The destination pointer will be released (if existing) and set to the source pointer. The source pointer will be set to 0.

**Syntax**

Definition:
FUNCTION F_VN_ConvertITcUnknownToITcVnBitmapExport : HRESULT
VAR_INPUT
    ipSrc : Reference To ITcUnknown;
    ipDest : Reference To ITcVnBitmapExport;
    hrPrev : HRESULT;
END_VAR

## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrc</td>
<td>Reference To ITcUnknown [399]</td>
<td>Source pointer</td>
</tr>
<tr>
<td>ipDest</td>
<td>Reference To ITcVnBitmapExport [371]</td>
<td>Destination pointer</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

## Required License

TC3 Vision Base

## System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.19.5 F_VN_ConvertITcUnknownToITcVnContainer

Convert an ITcUnknown interface pointer to an ITcVnContainer interface pointer. The destination pointer will be released (if existing) and set to the source pointer. The source pointer will be set to 0.

### Syntax

**Definition:**

FUNCTION F_VN_ConvertITcUnknownToITcVnContainer : HRESULT
VAR_INPUT
    ipSrc : Reference To ITcUnknown;
    ipDest : Reference To ITcVnContainer;
    hrPrev : HRESULT;
END_VAR
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrc</td>
<td>Reference To ITcUnknown</td>
<td>Source pointer</td>
</tr>
<tr>
<td>ipDest</td>
<td>Reference To ITcVnImage</td>
<td>Destination pointer</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.19.6 F_VN_ConvertITcUnknownToITcVnImage

F_VN_ConvertITcUnknownToITcVnImage

ipSrc  Reference To ITcUnknown  
ipDest Reference To ITcVnImage  
hrPrev HRESULT

Convert an ITcUnknown interface pointer to an ITcVnImage interface pointer. The destination pointer will be released (if existing) and set to the source pointer. The source pointer will be set to 0.

### Syntax

**Definition:**

```plaintext
FUNCTION F_VN_ConvertITcUnknownToITcVnImage : HRESULT
VAR_INPUT
    ipSrc  : Reference To ITcUnknown;
    ipDest : Reference To ITcVnImage;
    hrPrev : HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrc</td>
<td>Reference To ITcUnknown</td>
<td>Source pointer</td>
</tr>
<tr>
<td>ipDest</td>
<td>Reference To ITcVnImage</td>
<td>Destination pointer</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.19.7 F_VN_ConvertITcUnknownToITcVnMlModel

F_VN_ConvertITcUnknownToITcVnMlModel

ipSrc Reference To ITcUnknown

ipDest Reference To ITcVnMlModel

hrPrev HRESULT

Convert an ITcUnknown interface pointer to an ITcVnMlModel interface pointer. The destination pointer will be released (if existing) and set to the source pointer. The source pointer will be set to 0.

Syntax

Definition:

FUNCTION F_VN_ConvertITcUnknownToITcVnMlModel : HRESULT

VAR_INPUT

  ipSrc : Reference To ITcUnknown;
  ipDest : Reference To ITcVnMlModel;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipSrc</td>
<td>Reference To ITcUnknown [399]</td>
<td>Source pointer</td>
</tr>
<tr>
<td>ipDest</td>
<td>Reference To ITcVnMlModel [392]</td>
<td>Destination pointer</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.19.8  F_VN_GetTimestamp

Get the timestamp of an object (e.g. ITcVnImage or ITcVnContainer)

Syntax

Definition:

FUNCTION F_VN_GetTimestamp : HRESULT
VAR_INPUT
 ipUnknown : ITCUnknown;
 nTimestamp : Reference To LINT;
 hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipUnknown</td>
<td>ITCUnknown [399]</td>
<td>An object implementing ITCUnknown and ITcVnTimestamp (e.g. ITcVnImage or ITcVnContainer)</td>
</tr>
<tr>
<td>nTimestamp</td>
<td>Reference To LINT</td>
<td>Returns the timestamp</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.19.9  F_VN_HuMomentInvariants

Computes the Hu moment invariants.

Syntax

Definition:

FUNCTION F_VN_HuMomentInvariants : HRESULT
VAR_INPUT
 stMoments : Reference To TcVnMoments;
 aHuInvariants : Reference To TcVnArray7_LREAL;
 hrPrev : HRESULT;
END_VAR
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stMoments</td>
<td>Reference To</td>
<td>Moments from which to compute the Hu invariants</td>
</tr>
<tr>
<td>aHuInvariants</td>
<td>Reference To TcVnMoments [209]</td>
<td>Returns the Hu moment invariants</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.19.10 **F_VN_InitMatrixStruct**

```
FUNCTION F_VN_InitMatrixStruct : HRESULT
VAR_INPUT
    pSrcBuffer   : PVOID;
    stDestMatrix : Reference To TcVnMatrix;
    nRows        : UDINT;
    nCols        : UDINT;
    eElementType : ETcVnElementType;
    hrPrev       : HRESULT;
END_VAR
```

Initialize a struct of the type TcVnMatrix extending a buffer with metainformation so that it can be used as a matrix.

**Syntax**

**Definition:**

```
FUNCTION F_VN_InitMatrixStruct : HRESULT
VAR_INPUT
    pSrcBuffer   : PVOID;
    stDestMatrix : Reference To TcVnMatrix;
    nRows        : UDINT;
    nCols        : UDINT;
    eElementType : ETcVnElementType;
    hrPrev       : HRESULT;
END_VAR
```
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pSrcBuffer</td>
<td>PVOID</td>
<td>Source buffer</td>
</tr>
<tr>
<td>stDestMatrix</td>
<td>Reference To TcVnMatrix</td>
<td>Returns completed matrix struct</td>
</tr>
<tr>
<td>nRows</td>
<td>UDINT</td>
<td>Matrix rows</td>
</tr>
<tr>
<td>nCols</td>
<td>UDINT</td>
<td>Matrix columns</td>
</tr>
<tr>
<td>eElementType</td>
<td>ETcVnElementType</td>
<td>Type of the matrix elements</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Further information

The function F_VN_InitMatrixStruct creates a matrix structure of the type TcVnMatrix [208] from any memory area.

Parameter

Data buffer

The pointer pSrcBuffer must point to the memory area containing the data for the matrix. This could be an array, for example.

Matrix description

The matrix must be described by the number of rows nRows and columns nCols as well as by the data type of the element eElementType.

Result matrix

The matrix structure stDestMatrix of the type TcVnMatrix [208] is returned as the result.

Application

The creation of a $3 \times 3$ matrix looks like this, for example:

```pascal
VAR
  aMatrixArray : ARRAY [0..2, 0..2] OF USINT := [
    -1, 0, 1,
    -2, 0, 2,
    -1, 0, 1
  ];
END_VAR
hr := F_VN_InitMatrixStruct(
  pSrcBuffer := ADR(aMatrixArray),
  stDestMatrix := stMatrix,
  nRows := 3, // must match with array dimensions
  nCols := 3, // must match with array dimensions
  eElementType := TCVN_ET_USINT, // must match with array type
  hrPrev := hr
);
```

Required License

TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.19.11 F_VN_InvertMatrix3x3

**F_VN_InvertMatrix3x3**

- **aSrcMatrix** : Reference To TcVnMatrix3x3_LREAL
- **aInvertedMatrix** : Reference To TcVnMatrix3x3_LREAL
- **hrPrev** : HRESULT

Invert a 3x3 matrix.

**Syntax**

**Definition:**

FUNCTION F_VN_InvertMatrix3x3 : HRESULT
VAR_INPUT
  aSrcMatrix  : Reference To TcVnMatrix3x3_LREAL;
  aInvertedMatrix : Reference To TcVnMatrix3x3_LREAL;
  hrPrev       : HRESULT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aSrcMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>Source matrix</td>
</tr>
<tr>
<td>aInvertedMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL [153]</td>
<td>Returns inverted matrix</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.4.19.12 F_VN_LineIntersectionPoint

Computes the intersection point between two lines (returns S_FALSE if the provided lines are parallel).

**Syntax**

**Definition:**

```
FUNCTION F_VN_LineIntersectionPoint : HRESULT
VAR_INPUT
  aLine1 : Reference To TcVnVector4_LREAL;
  aLine2 : Reference To TcVnVector4_LREAL;
  aIntersecPoint : Reference To TcVnPoint2_REAL;
  hrPrev : HRESULT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aLine1</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>First line. The first and second element describe the x and y component of a vector collinear to the line. The third and fourth element describe the x and y component of a point on the line.</td>
</tr>
<tr>
<td>aLine2</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>Second line. The first and second element describe the x and y component of a vector collinear to the line. The third and fourth element describe the x and y component of a point on the line.</td>
</tr>
<tr>
<td>aIntersecPoint</td>
<td>Reference To TcVnPoint2_REAL</td>
<td>Returns the intersection point of aLine1 and aLine2.</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.19.13 F_VN_LineIntersectionPointAndAngle

**Syntax**

**Definition:**

```
FUNCTION F_VN_LineIntersectionPointAndAngle : HRESULT
VAR_INPUT
  aLine1 : Reference To TcVnVector4_LREAL;
  aLine2 : Reference To TcVnVector4_LREAL;
  aIntersecPoint : Reference To TcVnPoint2_REAL;
  fAngle : Reference To TcVnPoint2_REAL;
  bAngleInDegrees : BOOL;
  hrPrev : HRESULT;
END_VAR
```
Computes the intersection point and angle between two lines (returns S_FALSE if the provided lines are parallel).

**Syntax**

**Definition:**

FUNCTION F_VN_LineIntersectionPointAndAngle : HRESULT

VAR_INPUT

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aLine1</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>First line. The first and second element describe the x and y component of a vector collinear to the line. The third and fourth element describe the x and y component of a point on the line.</td>
</tr>
<tr>
<td>aLine2</td>
<td>Reference To TcVnVector4_LREAL</td>
<td>Second line. The first and second element describe the x and y component of a vector collinear to the line. The third and fourth element describe the x and y component of a point on the line.</td>
</tr>
<tr>
<td>aIntersecPoint</td>
<td>Reference To TcVnPoint2_REAL</td>
<td>Returns the intersection point of aLine1 and aLine2.</td>
</tr>
<tr>
<td>fAngle</td>
<td>Reference To REAL</td>
<td>Returns the intersection angle of aLine1 and aLine2.</td>
</tr>
<tr>
<td>bAngleInDegrees</td>
<td>BOOL</td>
<td>If TRUE, fAngle is in degrees. If FALSE, fAngle is in radians.</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.19.14 **F_VN_MultiplyMatrices**

Matrix multiplication of two matrices (A * B = C). All matrices are represented by structs holding a pointer to an array of the appropriate size.
Syntax

Definition:
FUNCTION F_VN_MultiplyMatrices : HRESULT
VAR_INPUT
  stSrcMatrix1 : Reference To TcVnMatrix;
  stSrcMatrix2 : Reference To TcVnMatrix;
  stDestMatrix : Reference To TcVnMatrix;
  hrPrev : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stSrcMatrix1</td>
<td>Reference To TcVnMatrix 208</td>
<td>First source matrix (A)</td>
</tr>
<tr>
<td>stSrcMatrix2</td>
<td>Reference To TcVnMatrix 208</td>
<td>Second source matrix (B)</td>
</tr>
<tr>
<td>stDestMatrix</td>
<td>Reference To TcVnMatrix 208</td>
<td>Destination matrix (C) (The destination matrix is filled by this function, but the required memory needs to be provided.)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT 135</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT 135

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.19.15  F_VN_RotatedRectangleCorners

Computes the 4 corner points of a rotated rectangle.

Syntax

Definition:
FUNCTION F_VN_RotatedRectangleCorners : HRESULT
VAR_INPUT
  stRect : Reference To TcVnRotatedRectangle;
  aCorners : Reference To TcVnArray4_Point2_REAL;
  hrPrev : HRESULT;
END_VAR

F_VN_RotatedRectangleCorners

stRect  Reference To TcVnRotatedRectangle
aCorners Reference To TcVnArray4_Point2_REAL
hrPrev HRESULT

F_VN_RotatedRectangleCorners
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stRect</td>
<td>Reference To TcVnRotatedRectangle</td>
<td>Rotated rectangle</td>
</tr>
<tr>
<td>aCorners</td>
<td>Reference To TcVnArray4Point2_REAL</td>
<td>Returns the 4 corner points</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.19.16 F_VN_RotatedRectangleIntersection

Finds the intersection of 2 rotated rectangles.

**Syntax**

**Definition:**

```plaintext
FUNCTION F_VN_RotatedRectangleIntersection : HRESULT
VAR_INPUT
    stRect1 : Reference To TcVnRotatedRectangle;
    stRect2 : Reference To TcVnRotatedRectangle;
    ipIntersection : Reference To ITcVnContainer;
    eIntersection : Reference To ETcVnRectangleIntersection;
    hrPrev : HRESULT;
END_VAR
```
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stRect1</td>
<td>Reference To TcVnRotatedRectangle [222]</td>
<td>Rotated rectangle 1</td>
</tr>
<tr>
<td>stRect2</td>
<td>Reference To TcVnRotatedRectangle [222]</td>
<td>Rotated rectangle 2</td>
</tr>
<tr>
<td>ipIntersection</td>
<td>Reference To ITcVnContainer [345]</td>
<td>Returns the intersection points (ContainerType_Vector_TcVnPoint2_REAL, non-zero interface pointers are reused)</td>
</tr>
<tr>
<td>elIntersection</td>
<td>Reference To ETcVnRectangleIntersection [198]</td>
<td>Returns the intersection type</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.4.19.17 F_VN_SetRngSeed

Sets the internal pseudo random number generator seed (intended for testing purposes only).

#### Syntax

**Definition:**

```c
FUNCTION F_VN_SetRngSeed : HRESULT
VAR_INPUT
   nSeed   : DINT;
   hrPrev  : HRESULT;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nSeed</td>
<td>DINT</td>
<td>seed (0 sets the generator back to its initial state)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>
API reference

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.4.19.18 F_VN_UpdateTimestamp

Update the timestamp of an object to the current time (e.g. ITcVnImage or ITcVnContainer)

Syntax

Definition:

FUNCTION F_VN_UpdateTimestamp : HRESULT
VAR_INPUT
    ipUnknown : ITcUnknown;
    hrPrev    : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipUnknown</td>
<td>ITcUnknown [399]</td>
<td>An object implementing ITcUnknown and ITcVnTimestamp (e.g. ITcVnImage or ITcVnContainer)</td>
</tr>
<tr>
<td>hrPrev</td>
<td>HRESULT [135]</td>
<td>HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.5 Function Blocks

Communication with GigE Vision cameras or with the file system as well as complex image processing functions is mapped by function blocks. Function blocks are identified by the prefix "FB_VN_".

Grouping of function blocks

All function blocks of the TwinCAT Vision library are sorted thematically into the following groups:

- Camera Register Access [ 1282]
- File Access [ 1298]
- Image Acquisition [ 1314]
- Image Processing [ 1353]

6.5.1 License Overview

The following list shows the assignment of TwinCAT Vision function blocks to available licenses (see license model [ 12]). It shows which license is required for the application of certain function blocks.

The basic license TC3 Vision Base (TF7100) is always required.

TC3 Vision Base

FB_VN_FileSourceControl [ 1316]
FB_VN_GevCameraControl [ 1324]
FB_VN_ReadCalibrationPattern [ 1301]
FB_VN_ReadContainer [ 1302]
FB_VN_ReadImage [ 1304]
FB_VN_ReadMemory [ 1283]
FB_VN_ReadMlModel [ 1306]
FB_VN_ReadRegister_REAL [ 1285]
FB_VN_ReadRegister_UDINT [ 1287]
FB_VN_ReadRegister_ULINT [ 1289]
FB_VN_SimpleCameraControl [ 1348]
FB_VN_WriteContainer [ 1308]
FB_VN_WriteImage [ 1310]
FB_VN_WriteMemory [ 1291]
FB_VN_WriteMlModel [ 1312]
FB_VN_WriteRegister_REAL [ 1293]
FB_VN_WriteRegister_UDINT [ 1295]
FB_VN_WriteRegister_ULINT [ 1297]
6.5.2 Camera Register Access

This group contains function blocks for reading and writing camera registers or memory areas.

Function blocks

UDINT-Register (4 Byte)
- FB_VN_ReadRegister_UDINT
- FB_VN_WriteRegister_UDINT

ULINT register (8 bytes)
- FB_VN_ReadRegister_ULINT
- FB_VN_WriteRegister_ULINT

REAL-Register (4 Byte)
- FB_VN_ReadRegister_REAL
- FB_VN_WriteRegister_REAL

Larger memory areas
- FB_VN_ReadMemory
- FB_VN_WriteMemory

Parameter

The following parameters work in a similar way with all function blocks in this group:

Address

The address `nAddress` specifies the register of the camera to be read or saved. The type `GVCP_REGISTER_ADDRESS` is merely an alias for UDINT; therefore, the address must be 4 bytes (or 32 bits) long. For larger memory areas of the camera (see `FB_VN_ReadMemory`) the address points to the beginning of the area to be read or saved.

Execute the function

The triggers for write and read commands are `bWrite` and `bRead` respectively. They are triggered by a rising edge (change from FALSE to TRUE). The relevant input parameters are saved internally so that a parameter change during the execution has no effect. A falling edge of the trigger likewise has no effect.

Calling function blocks

Please note that the description of parameters for function blocks only has an effect if the function blocks are actually called.

Timeout

So as not to have to wait a long time for the execution of the function block in the case of an unknown error, the maximum permitted execution time can be set with `nTimeout`. The execution will be aborted if this time is exceeded without a successful result after triggering the command trigger. The timeout error is then recognizable by the two outputs `bError` and `nErrorId`.

Working state

The output `bBusy` indicates whether the function block is busy. No new command can be triggered if `bBusy = TRUE`. 
By means of a suitable IF query it is possible to determine whether the execution of the function block is completed:

```
IF NOT fbReadOrWrite.bBusy THEN
    // Function block execution is complete or error occurred.
    // Next write-command can be issued.
END_IF
```

**Error state**

The error state is indicated by `bError` binarily indicating an error and `nErrorId` outputting the error code (ADS Return Code [1486]) if applicable. The error state is reset as soon as a new command is triggered. The error code is `PENDING` (16#71E) if the function block is busy (`bBusy = TRUE`).

The hexadecimal value of the error code can be extracted as follows:

```
IF fbReadOrWrite.bError THEN
    // Show relevant error code for debugging purposes:
    nErrorCode := fbReadOrWrite.nErrorId AND 16#FFF;
END_IF
```

**Examples**

- Reading/writing a register [1442]

### 6.5.2.1 FB_VN_ReadMemory

This FB reads consecutive memory locations from the camera. Requires an open control channel (e.g. by calling `FB_VN_GevCameraControl.OpenCamera()` before)

**Syntax**

**Definition:**

```
FUNCTION_BLOCK FB_VN_ReadMemory
VAR_INPUT
    nAddress : GVCP_REGISTER_ADDRESS;
    nLength  : UINT;
    pBuffer  : PVOID;
    bRead    : BOOL;
    nTimeout : TIME;
END_VAR
VAR_OUTPUT
    bBusy    : BOOL;
    bError   : BOOL;
    nErrorId : UDINT;
END_VAR
```
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nAddress</td>
<td>GVCP_REGISTER_ADDRESS [151]</td>
<td>S[151]</td>
<td>The address of the first byte to read from the camera (must be 32-bit aligned)</td>
</tr>
<tr>
<td>nLength</td>
<td>UINT</td>
<td></td>
<td>The number of bytes to read, starting with nAddress (must be a multiple of 4, not more than 536)</td>
</tr>
<tr>
<td>pBuffer</td>
<td>PVOID</td>
<td></td>
<td>Pointer to the buffer where the read memory content is written to (The buffer must have a size of at least nLength bytes!)</td>
</tr>
<tr>
<td>bRead</td>
<td>BOOL</td>
<td></td>
<td>Reading the memory is triggered by a rising edge at this input.</td>
</tr>
<tr>
<td>nTimeout</td>
<td>TIME</td>
<td>VISION_ADS_TIME_OUT</td>
<td>Indicates the time before the function is cancelled.</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bBusy</td>
<td>BOOL</td>
<td>This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.</td>
</tr>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrorId'. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.</td>
</tr>
<tr>
<td>nErrorId</td>
<td>UDINT</td>
<td>Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.</td>
</tr>
</tbody>
</table>

Further information

**Open camera Control Channel needed!**

The Control Channel of the GigE Vision camera must be open in order to be able to use this function block. In addition, some camera parameters can only be changed if the image acquisition is stopped. It is therefore recommended to place the camera in the state TCVN_CS_OPENED first.

### Parameter

**nAddress**

The register address identifies the camera register to be addressed. It is of the type GVCP_REGISTER_ADDRESS, which is an alias of the type UDINT. Thus, all register addresses have a size of 4 bytes. The register address of a camera parameter can be seen at the respective point in the configuration tree [75] of the GigE Vision camera instance:
**Required License**
TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.5.2.2 FB_VN_ReadRegister_REAL

**FB_VN_ReadRegister_REAL**

<table>
<thead>
<tr>
<th>nAddress</th>
<th>GVCP_REGISTER_ADDRESS</th>
<th>REAL</th>
<th>fValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>nEndian</td>
<td>UINT</td>
<td>BOOL</td>
<td>bBusy</td>
</tr>
<tr>
<td>bRead</td>
<td>BOOL</td>
<td>BOOL</td>
<td>bError</td>
</tr>
<tr>
<td>nTimeout</td>
<td>TIME</td>
<td>UDINT</td>
<td>nErrorId</td>
</tr>
</tbody>
</table>

This FB reads a REAL register from the camera. Requires an open control channel (e.g. by calling FB_VN_GevCameraControl.OpenCamera() before)

**Syntax**

**Definition:**

```c
FUNCTION_BLOCK FB_VN_ReadRegister_REAL
VAR_INPUT
    nAddress : GVCP_REGISTER_ADDRESS;
    nEndian : UINT;
    bRead : BOOL;
    nTimeout : TIME;
END_VAR
VAR_OUTPUT
    fValue : REAL;
    bBusy : BOOL;
    bError : BOOL;
    nErrorId : UDINT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nAddress</td>
<td>GVCP_REGISTER_ADDRESS</td>
<td>[151]</td>
<td>The address of the register that should be read</td>
</tr>
<tr>
<td>nEndian</td>
<td>UINT</td>
<td></td>
<td>OPTIONAL: The endianness of the register. 0 = Big, 1 = Little. Default: 0</td>
</tr>
<tr>
<td>bRead</td>
<td>BOOL</td>
<td></td>
<td>Reading the register is triggered by a rising edge at this input.</td>
</tr>
<tr>
<td>nTimeout</td>
<td>TIME</td>
<td>VISION_ADS_TIME_OUT</td>
<td>Indicates the time before the function is cancelled.</td>
</tr>
</tbody>
</table>
### Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fValue</td>
<td>REAL</td>
<td>The read value</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
<td>This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.</td>
</tr>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrorId'. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.</td>
</tr>
<tr>
<td>nErrorId</td>
<td>UDINT</td>
<td>Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.</td>
</tr>
</tbody>
</table>

### Further information

- **Open camera Control Channel needed!**
  The Control Channel of the GigE Vision camera must be open in order to be able to use this function block. In addition, some camera parameters can only be changed if the image acquisition is stopped. It is therefore recommended to place the camera in the state TCVN_CS_OPENED first.

### Parameter

- **nAddress**
  The register address identifies the camera register to be addressed. It is of the type GVCP_REGISTER_ADDRESS, which is an alias of the type UDINT. Thus, all register addresses have a size of 4 bytes. The register address of a camera parameter can be seen at the respective point in the configuration tree of the GigE Vision camera instance:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>PixelFormat</td>
</tr>
<tr>
<td>Value</td>
<td>BayerRG8</td>
</tr>
<tr>
<td>Address</td>
<td>0x00012120</td>
</tr>
<tr>
<td>Length</td>
<td>4 Bytes</td>
</tr>
</tbody>
</table>

- **nEndian**
  Endianness describes the order in which the bytes of a camera register are arranged. This does not need to be considered any further in terms of the application and can be adopted as a property of the respective parameter of the configuration tree of the GigE Vision camera instance:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>4 Bytes</td>
</tr>
<tr>
<td>Endianness</td>
<td>Big</td>
</tr>
<tr>
<td>Sign</td>
<td>Unsigned</td>
</tr>
</tbody>
</table>

  A Big Endian is represented by nEndian := 0 and a Small Endian by nEndian := 1. Big Endian is the default.

### Samples

- Reading/writing a register [1442]
Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.5.2.3 FB_VN_ReadRegister_UDINT

This FB reads a DUINT register from the camera. Requires an open control channel (e.g. by calling FB_VN_GevCameraControl.OpenCamera() before)

Syntax

Definition:

```
FUNCTION_BLOCK FB_VN_ReadRegister_UDINT
VAR_INPUT
    nAddress : GVCP_REGISTER_ADDRESS;
    nEndian : UINT;
    bRead : BOOL;
    nTimeout : TIME;
END_VAR
VAR_OUTPUT
    nValue : UDINT;
    bBusy : BOOL;
    bError : BOOL;
    nErrorId : UDINT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nAddress</td>
<td>GVCP_REGISTER_ADDRESS</td>
<td>151</td>
<td>The address of the register that should be read</td>
</tr>
<tr>
<td>nEndian</td>
<td>UINT</td>
<td>Optional: The endianness of the register. 0 = Big, 1 = Little. Default: 0</td>
<td></td>
</tr>
<tr>
<td>bRead</td>
<td>BOOL</td>
<td></td>
<td>Reading the register is triggered by a rising edge at this input.</td>
</tr>
<tr>
<td>nTimeout</td>
<td>TIME</td>
<td>VISION_ADS_TIMEOUT</td>
<td>Indicates the time before the function is cancelled.</td>
</tr>
</tbody>
</table>
### Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nValue</td>
<td>UDINT</td>
<td>The read value</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
<td>This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.</td>
</tr>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrorId'. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.</td>
</tr>
<tr>
<td>nErrorId</td>
<td>UDINT</td>
<td>Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.</td>
</tr>
</tbody>
</table>

### Further information

- **Open camera Control Channel needed!**
  The Control Channel of the GigE Vision camera must be open in order to be able to use this function block. In addition, some camera parameters can only be changed if the image acquisition is stopped. It is therefore recommended to place the camera in the state TCVN_CS_OPENED first.

### Parameter

- **nAddress**
  The register address identifies the camera register to be addressed. It is of the type GVCP_REGISTER_ADDRESS, which is an alias of the type UDINT. Thus, all register addresses have a size of 4 bytes. The register address of a camera parameter can be seen at the respective point in the configuration tree of the GigE Vision camera instance:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>PixelFormat</td>
</tr>
<tr>
<td>Value</td>
<td>BayerRG8</td>
</tr>
<tr>
<td>Address</td>
<td>0x00012120</td>
</tr>
<tr>
<td>Length</td>
<td>4 Bytes</td>
</tr>
</tbody>
</table>

- **nEndian**
  Endianness describes the order in which the bytes of a camera register are arranged. This does not need to be considered any further in terms of the application and can be adopted as a property of the respective parameter of the configuration tree of the GigE Vision camera instance:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>4 Bytes</td>
</tr>
<tr>
<td>Endianness</td>
<td>Big</td>
</tr>
<tr>
<td>Sign</td>
<td>Unsigned</td>
</tr>
</tbody>
</table>

  A Big Endian is represented by nEndian := 0 and a Small Endian by nEndian := 1. Big Endian is the default.

### Samples

- Reading/writing a register

1288  
Version: 1.3  
TF7000 - TF7300
Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.5.2.4  **FB_VN_ReadRegister_ULINT**

This FB reads a ULINT register from the camera. Requires an open control channel (e.g. by calling FB_VN_GevCameraControl.OpenCamera() before)

**Syntax**

Definition:

```plaintext
FUNCTION_BLOCK FB_VN_ReadRegister_ULINT
VAR_INPUT
   nAddress : GVCP_REGISTER_ADDRESS;
   nEndian : UINT;
   bRead : BOOL;
   nTimeout : TIME;
END_VAR
VAR_OUTPUT
   nValue : ULINT;
   bBusy : BOOL;
   bError : BOOL;
   nErrorId : UDINT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nAddress</td>
<td>GVCP_REGISTER_ADDRESS</td>
<td></td>
<td>The address of the register that should be read</td>
</tr>
<tr>
<td>nEndian</td>
<td>UINT</td>
<td></td>
<td>OPTIONAL: The endianness of the register. 0 = Big, 1 = Little. Default: 0</td>
</tr>
<tr>
<td>bRead</td>
<td>BOOL</td>
<td></td>
<td>Reading the register is triggered by a rising edge at this input.</td>
</tr>
<tr>
<td>nTimeout</td>
<td>TIME</td>
<td>VISION_ADS_TIMEOUT</td>
<td>Indicates the time before the function is cancelled.</td>
</tr>
</tbody>
</table>
## Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nValue</td>
<td>ULINT</td>
<td>The read value</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
<td>This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.</td>
</tr>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrorId'. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.</td>
</tr>
<tr>
<td>nErrorId</td>
<td>UDINT</td>
<td>Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.</td>
</tr>
</tbody>
</table>

### Further information

**Open camera Control Channel needed!**

The Control Channel of the GigE Vision camera must be open in order to be able to use this function block. In addition, some camera parameters can only be changed if the image acquisition is stopped. It is therefore recommended to place the camera in the state TCVN_CS_OPENED first.

### Parameter

**nAddress**

The register address identifies the camera register to be addressed. It is of the type GVCP_REGISTER_ADDRESS, which is an alias of the type UDINT. Thus, all register addresses have a size of 4 bytes. The register address of a camera parameter can be seen at the respective point in the configuration tree of the GigE Vision camera instance:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>PixelFormat</td>
</tr>
<tr>
<td>Value</td>
<td>BayerRG8</td>
</tr>
<tr>
<td>Address</td>
<td>0x00012120</td>
</tr>
<tr>
<td>Length</td>
<td>4 Bytes</td>
</tr>
</tbody>
</table>

**nEndian**

Endianess describes the order in which the bytes of a camera register are arranged. This does not need to be considered any further in terms of the application and can be adopted as a property of the respective parameter of the configuration tree of the GigE Vision camera instance:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>4 Bytes</td>
</tr>
<tr>
<td>Endianness</td>
<td>Big</td>
</tr>
<tr>
<td>Sign</td>
<td>Unsigned</td>
</tr>
</tbody>
</table>

A Big Endian is represented by nEndian := 0 and a Small Endian by nEndian := 1. Big Endian is the default.

### Examples

Reading/writing a register [1442]
6.5.2.5 FB_VN_WriteMemory

This FB writes data to consecutive memory locations on the camera. Requires an open control channel (e.g. by calling FB_VN_GevCameraControl.OpenCamera() before).

Syntax

Definition:

```plaintext
FUNCTION_BLOCK FB_VN_WriteMemory
VAR_INPUT
  nAddress : GVCP_REGISTER_ADDRESS;
  nLength : UINT;
  pData : PVOID;
  bWrite : BOOL;
  nTimeout : TIME;
END_VAR
VAR_OUTPUT
  bBusy : BOOL;
  bError : BOOL;
  nErrorId : UDINT;
  nBytesWritten : UINT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nAddress</td>
<td>GVCP_REGISTER_ADDRESS</td>
<td></td>
<td>The address of the first byte to write to the camera (must be 32-bit aligned)</td>
</tr>
<tr>
<td>nLength</td>
<td>UINT</td>
<td></td>
<td>The number of bytes to write, starting at nAddress (must be a multiple of 4, not more than 536)</td>
</tr>
<tr>
<td>pData</td>
<td>PVOID</td>
<td></td>
<td>Pointer to the data that should be written to the camera (must have a size of at least nLength bytes!)</td>
</tr>
<tr>
<td>bWrite</td>
<td>BOOL</td>
<td></td>
<td>Writing the memory is triggered by a rising edge at this input.</td>
</tr>
<tr>
<td>nTimeout</td>
<td>TIME</td>
<td>VISION_ADS_TIME_OUT</td>
<td>Indicates the time before the function is cancelled.</td>
</tr>
</tbody>
</table>
### Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bBusy</td>
<td>BOOL</td>
<td>This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.</td>
</tr>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrorId'. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.</td>
</tr>
<tr>
<td>nErrorId</td>
<td>UDINT</td>
<td>Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.</td>
</tr>
<tr>
<td>nBytesWritten</td>
<td>UINT</td>
<td>The number of bytes that have been successfully written to the camera.</td>
</tr>
</tbody>
</table>

### Further information

**Open camera Control Channel needed!**

The Control Channel of the GigE Vision camera must be open in order to be able to use this function block. In addition, some camera parameters can only be changed if the image acquisition is stopped. It is therefore recommended to place the camera in the state TCVN_CS_OPENED first.

### Parameter

**nAddress**

The register address identifies the camera register to be addressed. It is of the type GVCP_REGISTER_ADDRESS, which is an alias of the type UDINT. Thus, all register addresses have a size of 4 bytes. The register address of a camera parameter can be seen at the respective point in the configuration tree [75] of the GigE Vision camera instance:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>PixelFormat</td>
</tr>
<tr>
<td>Value</td>
<td>BayerRG8</td>
</tr>
<tr>
<td>Address</td>
<td>0x00012120</td>
</tr>
<tr>
<td>Length</td>
<td>4 Bytes</td>
</tr>
</tbody>
</table>

### Required License

TC3 Vision Base

### System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.5.2.6  FB_VN_WriteRegister_REAL

**FUNCTION_BLOCK FB_VN_WriteRegister_REAL**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nAddress</td>
<td>GVCP_REGISTER_ADDRESS</td>
<td></td>
<td>The address of the register that should be written</td>
</tr>
<tr>
<td>fValue</td>
<td>REAL</td>
<td></td>
<td>The value to write</td>
</tr>
<tr>
<td>nEndian</td>
<td>UINT</td>
<td></td>
<td>OPTIONAL: The endianness of the register. 0 = Big, 1 = Little. Default: 0</td>
</tr>
<tr>
<td>bWrite</td>
<td>BOOL</td>
<td></td>
<td>Writing the register is triggered by a rising edge at this input.</td>
</tr>
<tr>
<td>nTimeout</td>
<td>TIME</td>
<td>VISION_ADS_TIMEOUT</td>
<td>Indicates the time before the function is cancelled.</td>
</tr>
</tbody>
</table>

**Syntax**

**Definition:**

```plaintext
FUNCTION_BLOCK FB_VN_WriteRegister_REAL
VAR_INPUT
    nAddress : GVCP_REGISTER_ADDRESS;
    fValue : REAL;
    nEndian : UINT;
    bWrite : BOOL;
    nTimeout : TIME;
END_VAR
VAR_OUTPUT
    bBusy : BOOL;
    bError : BOOL;
    nErrorId : UDINT;
END_VAR
```

**Inputs**

**Outputs**

This FB writes a REAL value into a register on the camera. Requires an open control channel (e.g. by calling FB_VN_GevCameraControl.OpenCamera() before)
Further information

Open camera Control Channel needed!
The Control Channel of the GigE Vision camera must be open in order to be able to use this function block. In addition, some camera parameters can only be changed if the image acquisition is stopped. It is therefore recommended to place the camera in the state TCVN_CS_OPENED first.

Parameter

nAddress
The register address identifies the camera register to be addressed. It is of the type GVCP_REGISTER_ADDRESS, which is an alias of the type UDINT. Thus, all register addresses have a size of 4 bytes. The register address of a camera parameter can be seen at the respective point in the configuration tree of the GigE Vision camera instance:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>PixelFormat</td>
</tr>
<tr>
<td>Value</td>
<td>BayerRG8</td>
</tr>
<tr>
<td><strong>Address</strong></td>
<td><strong>0x00012120</strong></td>
</tr>
<tr>
<td>Length</td>
<td>4 Bytes</td>
</tr>
</tbody>
</table>

nEndian
Endianess describes the order in which the bytes of a camera register are arranged. This does not need to be considered any further in terms of the application and can be adopted as a property of the respective parameter of the configuration tree of the GigE Vision camera instance:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>4 Bytes</td>
</tr>
<tr>
<td><strong>Endianess</strong></td>
<td><strong>Big</strong></td>
</tr>
<tr>
<td>Sign</td>
<td>Unsigned</td>
</tr>
</tbody>
</table>

A Big Endian is represented by nEndian := 0 and a Small Endian by nEndian := 1. Big Endian is the default.

Samples

• Reading/writing a register [1442]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.5.2.7  FB_VN_WriteRegister_UDINT

This FB writes a DINT value into a register on the camera. Requires an open control channel (e.g. by calling FB_VN_GevCameraControl.OpenCamera() before)

**Syntax**

**Definition:**

FUNCTION BLOCK FB_VN_WriteRegister_UDINT

VAR_INPUT
  nAddress : GVCP_REGISTER_ADDRESS;
  nValue : UDINT;
  nEndian : UINT;
  bWrite : BOOL;
  nTimeout : TIME;
END_VAR

VAR_OUTPUT
  bBusy : BOOL;
  bError : BOOL;
  nErrorId : UDINT;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nAddress</td>
<td>GVCP REGISTER ADDRESS S[1..151]</td>
<td></td>
<td>The address of the register that should be written</td>
</tr>
<tr>
<td>nValue</td>
<td>UDINT</td>
<td></td>
<td>The value to write</td>
</tr>
<tr>
<td>nEndian</td>
<td>UINT</td>
<td></td>
<td>OPTIONAL: The endianness of the register, 0 = Big, 1 = Little. Default: 0</td>
</tr>
<tr>
<td>bWrite</td>
<td>BOOL</td>
<td></td>
<td>Reading the register is triggered by a rising edge at this input.</td>
</tr>
<tr>
<td>nTimeout</td>
<td>TIME</td>
<td>VISION_ADS_TIME OUT</td>
<td>Indicates the time before the function is cancelled.</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bBusy</td>
<td>BOOL</td>
<td>This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.</td>
</tr>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrorId'. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.</td>
</tr>
<tr>
<td>nErrorId</td>
<td>UDINT</td>
<td>Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.</td>
</tr>
</tbody>
</table>
Further information

Open camera Control Channel needed!

The Control Channel of the GigE Vision camera must be open in order to be able to use this function block. In addition, some camera parameters can only be changed if the image acquisition is stopped. It is therefore recommended to place the camera in the state TCVN_CS_OPENED first.

Parameter

nAddress

The register address identifies the camera register to be addressed. It is of the type GVCP_REGISTER_ADDRESS, which is an alias of the type UDINT. Thus, all register addresses have a size of 4 bytes. The register address of a camera parameter can be seen at the respective point in the configuration tree of the GigE Vision camera instance:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>PixelFormat</td>
</tr>
<tr>
<td>Value</td>
<td>BayerRG8</td>
</tr>
<tr>
<td><strong>Address</strong></td>
<td>0x00012120</td>
</tr>
<tr>
<td>Length</td>
<td>4 Bytes</td>
</tr>
</tbody>
</table>

nEndian

Endianess describes the order in which the bytes of a camera register are arranged. This does not need to be considered any further in terms of the application and can be adopted as a property of the respective parameter of the configuration tree of the GigE Vision camera instance:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>4 Bytes</td>
</tr>
<tr>
<td><strong>Endianness</strong></td>
<td>Big</td>
</tr>
<tr>
<td>Sign</td>
<td>Unsigned</td>
</tr>
</tbody>
</table>

A Big Endian is represented by nEndian := 0 and a Small Endian by nEndian := 1. Big Endian is the default.

Samples

- Reading/writing a register [1442]

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
6.5.2.8  FB_VN_WriteRegister_ULINT

This FB writes a U|INT value into a register on the camera. Requires an open control channel (e.g. by calling FB_VN_GevCameraControl.OpenCamera() before)

Syntax

**Definition:**

```plaintext
FUNCTION_BLOCK FB_VN_WriteRegister_ULINT
VAR_INPUT
  nAddress : GVCP_REGISTER_ADDRESS;
  nValue : ULINT;
  nEndian : UINT;
  bWrite : BOOL;
  nTimeout : TIME;
END_VAR
VAR_OUTPUT
  bBusy : BOOL;
  bError : BOOL;
  nErrorId : UDINT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nAddress</td>
<td>GVCP_REGISTER_ADDRESS</td>
<td></td>
<td>The address of the register that should be written</td>
</tr>
<tr>
<td>nValue</td>
<td>ULINT</td>
<td></td>
<td>The value to write</td>
</tr>
<tr>
<td>nEndian</td>
<td>UINT</td>
<td>OPTIONAL: The endianness of the register. 0 = Big, 1 = Little. Default: 0</td>
<td></td>
</tr>
<tr>
<td>bWrite</td>
<td>BOOL</td>
<td></td>
<td>Reading the register is triggered by a rising edge at this input.</td>
</tr>
<tr>
<td>nTimeout</td>
<td>TIME</td>
<td>VISION_ADS_TIMEOUT</td>
<td>Indicates the time before the function is cancelled.</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bBusy</td>
<td>BOOL</td>
<td>This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.</td>
</tr>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrorId'. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.</td>
</tr>
<tr>
<td>nErrorId</td>
<td>UDINT</td>
<td>Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.</td>
</tr>
</tbody>
</table>
Further information

Open camera Control Channel needed!
The Control Channel of the GigE Vision camera must be open in order to be able to use this function block. In addition, some camera parameters can only be changed if the image acquisition is stopped. It is therefore recommended to place the camera in the state TCVN_CS_OPENED first.

Parameter

nAddress
The register address identifies the camera register to be addressed. It is of the type GVCP_REGISTER_ADDRESS, which is an alias of the type UDINT. Thus, all register addresses have a size of 4 bytes. The register address of a camera parameter can be seen at the respective point in the configuration tree of the GigE Vision camera instance:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>PixelFormat</td>
</tr>
<tr>
<td>Value</td>
<td>BayerRG8</td>
</tr>
<tr>
<td>Address</td>
<td>0x00012120</td>
</tr>
<tr>
<td>Length</td>
<td>4 Bytes</td>
</tr>
</tbody>
</table>

nEndian
Endianess describes the order in which the bytes of a camera register are arranged. This does not need to be considered any further in terms of the application and can be adopted as a property of the respective parameter of the configuration tree of the GigE Vision camera instance:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>4 Bytes</td>
</tr>
<tr>
<td>Endianess</td>
<td>Big</td>
</tr>
<tr>
<td>Sign</td>
<td>Unsigned</td>
</tr>
</tbody>
</table>

A Big Endian is represented by $n\text{Endian} := 0$ and a Small Endian by $n\text{Endian} := 1$. Big Endian is the default.

Examples

Reading/writing a register [1442]

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.5.3 File Access
This group contains function blocks for reading and writing data on the file system.
Function blocks

Images
- FB_VN_ReadImage [1304]
- FB_VN_WriteImage [1310]

Container
- FB_VN_ReadContainer [1302]
- FB_VN_WriteContainer [1308]

Machine learning Modelle
- FB_VN_ReadMIModel [1306]
- FB_VN_WriteMIModel [1312]

Calibration pattern
- FB_VN_ReadCalibrationPattern [1301]

General
These function blocks work asynchronously; the time from the first execution to the desired result extends over several cycles. The actual duration of each read/write operation depends on several factors, such as the size of the image and the behavior of the operating system. The following parameters are the same for all function blocks and are therefore described here once centrally.

Parameter

File path
The file path sFilePath specifies where to save a file or which file to load. Absolute and relative paths can be used. In the case of relative paths, the default paths set in the Service Configuration [63] are used as the basis.

The file path is stored internally on a rising edge of the write command bWrite so that changes to the path in later cycles have no effect.

Execute the function
The triggers for write and read commands are bWrite and bRead respectively. They are triggered by a rising edge (change from FALSE to TRUE). The relevant input parameters are saved internally so that a parameter change during the execution has no effect. A falling edge of the trigger likewise has no effect.

Calling function blocks
Please note that the description of parameters for function blocks only has an effect if the function blocks are actually called.

Timeout
So as not to have to wait a long time for the execution of the function block in the case of an unknown error, the maximum permitted execution time can be set with nTimeout. The execution will be aborted if this time is exceeded without a successful result after triggering the command trigger. The timeout error is then recognizable by the two outputs bError and nErrorId.

Working state
The output bBusy indicates whether the function block is busy. No new command can be triggered if bBusy = TRUE.

By means of a suitable IF query it is possible to determine whether the execution of the function block is completed:
Error state

The error state is indicated by `bError` binarily indicating an error and `nErrorId` outputting the error code (ADS Return Code [1486]) if applicable. The error state is reset as soon as a new command is triggered. The error code is `PENDING` (16#71E) if the function block is busy (`bBusy = TRUE`).

```
IF NOT fbReadOrWrite.bBusy THEN
    // Function block execution is complete or error occurred.
    // Next write-command can be issued.
END_IF
```

The hexadecimal value of the error code can be extracted as follows:

```
IF fbReadOrWrite.bError THEN
    // Show relevant error code for debugging purposes:
    nErrorCode := fbReadOrWrite.nErrorId AND 16#FFF;
END_IF
```

The most frequent error codes are:

<table>
<thead>
<tr>
<th>Hex</th>
<th>Dec</th>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>16#006</td>
<td>6</td>
<td>TARGETPORTNOTFOUND</td>
<td>ADS server not reachable. Essentially occurs when the TwinCAT Vision Service has not started. See Vision Service doesn't start [1471].</td>
</tr>
<tr>
<td>16#700</td>
<td>1792</td>
<td>ERROR</td>
<td>General error if saving or loading fails. This usually indicates that the specified file format is not supported or that the file path is unreachable. The error usually occurs in the TwinCAT Vision Service [63].</td>
</tr>
<tr>
<td>16#70B</td>
<td>1803</td>
<td>INVALIDPARM</td>
<td>Invalid parameter. This usually indicates that the interface pointer [131] is invalid.</td>
</tr>
<tr>
<td>16#719</td>
<td>1817</td>
<td>TIMEOUT</td>
<td>Timeout. Loading or saving has exceeded the maximum execution time <code>nTimeout</code>. In general, this occurs mainly with a heavy load on ADS.</td>
</tr>
<tr>
<td>16#71E</td>
<td>1822</td>
<td>PENDING</td>
<td>Not really an error, therefore the output <code>bError</code> is not set here. This code signals that the function block is busy. It occurs in parallel to the output <code>bBusy</code> if a new rising edge is registered while the function block is busy.</td>
</tr>
</tbody>
</table>

Examples

- Saving images from the PLC [1444]
6.5.3.1 FB_VN_ReadCalibrationPattern

This FB reads calibration pattern reference points from an xml file on the target pc.

Syntax

Definition:

```
FUNCTION_BLOCK FB_VN_ReadCalibrationPattern
VAR_INPUT
  sFilePath : STRING;
  ipDestContainer : Reference To ITcVnContainer;
  bRead : BOOL;
  nTimeout : TIME;
END_VAR
VAR_OUTPUT
  bBusy : BOOL;
  bError : BOOL;
  nErrorId : UDINT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sFilePath</td>
<td>STRING</td>
<td></td>
<td>Full path of the file or relative path to the default directory on the target pc (%TWINCAT3DIR%\CustomConfig\Vision\CalibrationPattern)</td>
</tr>
<tr>
<td>ipDestContainer</td>
<td>Reference To ITcVnContainer</td>
<td></td>
<td>Returns a container with the pattern points (ContainerType_Vector_TcVnPoint3_REAL)</td>
</tr>
<tr>
<td>bRead</td>
<td>BOOL</td>
<td></td>
<td>Reading the file is triggered by a rising edge at this input.</td>
</tr>
<tr>
<td>nTimeout</td>
<td>TIME</td>
<td>VISION_ADS_TIME</td>
<td>Indicates the time before the function is cancelled.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bBusy</td>
<td>BOOL</td>
<td>This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.</td>
</tr>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrorId'. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.</td>
</tr>
<tr>
<td>nErrorId</td>
<td>UDINT</td>
<td>Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.</td>
</tr>
</tbody>
</table>
Further information

The function block `FB_VN_ReadCalibrationPattern` loads the description of an individual calibration pattern [114] asynchronously from a file.

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g.</td>
<td>Tc3_Vision</td>
</tr>
<tr>
<td></td>
<td>Intel 4-core Atom CPU</td>
<td></td>
</tr>
</tbody>
</table>

6.5.3.2 FB_VN_ReadContainer

This FB reads a container from an xml file on the target pc.

Syntax

Definition:

```plaintext
FUNCTION_BLOCK FB_VN_ReadContainer
VAR_INPUT
  sFilePath : STRING;
  ipDestContainer : Reference To ITcVnContainer;
  nDestTypeGuid : GUID;
  bRead : BOOL;
  nTimeout : TIME;
END_VAR
VAR_OUTPUT
  bBusy : BOOL;
  bError : BOOL;
  nErrorId : UDINT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sFilePath</td>
<td>STRING</td>
<td></td>
<td>Full path of the file or relative path to the default directory on the target pc</td>
</tr>
<tr>
<td>ipDestContainer</td>
<td>Reference To ITcVnContainer</td>
<td></td>
<td>Returns a container with the loaded content</td>
</tr>
<tr>
<td>nDestTypeGuid</td>
<td>GUID</td>
<td></td>
<td>The type id of the container to return. If this parameter is not set, a suitable container type will be determined automatically (except for csv files).</td>
</tr>
<tr>
<td>bRead</td>
<td>BOOL</td>
<td></td>
<td>Reading the file is triggered by a rising edge at this input.</td>
</tr>
<tr>
<td>nTimeout</td>
<td>TIME</td>
<td>VISION_ADS_TIMEOUT</td>
<td>Indicates the time before the function is cancelled.</td>
</tr>
</tbody>
</table>
## Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bBusy</td>
<td>BOOL</td>
<td>This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.</td>
</tr>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrorId'. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.</td>
</tr>
<tr>
<td>nErrorId</td>
<td>UDINT</td>
<td>Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.</td>
</tr>
</tbody>
</table>

### Further information

The function block **FB_VN_ReadContainer** loads a container [143] from a file asynchronously into the PLC.

**Loading of modified containers.**

When loading containers, the vision service caches the data internally. Loading changed containers is therefore not possible in the standard system. To do this, adjust the cache settings [63] or see the example **Serial loading of images into the PLC** [1445] for an alternative.

### Parameter

Some parameters are described centrally in the section **File Access Parameters** [1299].

### Container type

The **container type** [151] of the container to be loaded can optionally be specified.

### Loaded container (Return value)

The loaded container is reachable via the interface pointer **ipDestContainer** following successful reading.

### Application

Loading a container with explicit type specification **ContainerType_Vector_TcVnPoint2_DINT** from the path **C:\TcVision\Container.xml** looks like this, for example:

```plaintext
fbReadContainer(
    sFilePath := 'C:\TcVision\Container.xml',
    ipDestContainer := ipContainer,
    nDestTypeGuid := ContainerType_Vector_TcVnPoint2_DINT,
    bRead := TRUE,
    nTimeout := T#500MS
);
```

IF NOT fbReadContainer.bBusy AND NOT fbReadContainer.bError THEN
   // container is accessible in ipContainer
END_IF

Related function block: **FB_VN_WriteContainer** [1308].

### Required License

TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.5.3.3 FB_VN_ReadImage

This FB reads an image from a file on the target pc.

Syntax

Definition:

```
FUNCTION_BLOCK FB_VN_ReadImage
VAR_INPUT
  sFilePath : STRING;
  ipDestImage : Reference To ITcVnImage;
  bRead : BOOL;
  nTimeout : TIME;
END_VAR
VAR_OUTPUT
  bBusy : BOOL;
  bError : BOOL;
  nErrorId : UDINT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sFilePath</td>
<td>STRING</td>
<td></td>
<td>Full path of the file or relative path to the default image directory on the target pc</td>
</tr>
<tr>
<td>ipDestImage</td>
<td>Reference To ITcVnImage</td>
<td></td>
<td>Returns an image with the loaded content</td>
</tr>
<tr>
<td>bRead</td>
<td>BOOL</td>
<td></td>
<td>Reading the file is triggered by a rising edge at this input.</td>
</tr>
<tr>
<td>nTimeout</td>
<td>TIME</td>
<td>VISION_ADS_TIME_OUT</td>
<td>Indicates the time before the function is cancelled.</td>
</tr>
</tbody>
</table>
## Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bBusy</td>
<td>BOOL</td>
<td>This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.</td>
</tr>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrorId'. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.</td>
</tr>
<tr>
<td>nErrorId</td>
<td>UDINT</td>
<td>Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.</td>
</tr>
</tbody>
</table>

### Further information

The function block **FB_VN_ReadImage** loads images [140] from a file asynchronously into the PLC.

#### Loading of modified images.

When loading images, the Vision service caches the data internally. Loading changed images is therefore not possible in the standard system. To do this, adjust the cache settings [63] or see the example Serial loading of images into the PLC [1445] for an alternative.

### Parameter

Some parameters are described centrally in the section File Access Parameters [1299].

### Destination image (Return value)

The loaded image is reachable via the interface pointer `ipDestImage` following successful reading.

### Application

For example, loading an image from the path `C:\TcVision\Image.bmp` looks like this:

```plaintext
fbReadImage(
   sFilePath := 'C:\TcVision\Image.bmp',
   ipDestImage := ipImageIn,
   bRead := TRUE,
   nTimeout := T#500MS
);

IF NOT fbReadImage.bBusy AND NOT fbReadImage.bError THEN
   // image is accessible in ipImageIn
END_IF
```

### Examples

- Serial loading of images into the PLC [1445]

### Related function block

- **FB_VN_WriteImage** [1310] for saving images.

### Required License

- TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g.</td>
<td>Tc3_Vision</td>
</tr>
<tr>
<td></td>
<td>Intel 4-core Atom CPU</td>
<td></td>
</tr>
</tbody>
</table>

6.5.3.4 FB_VN_ReadMlModel

This FB reads a machine learning model from a file on the target pc.

Syntax

Definition:

```plaintext
FUNCTION_BLOCK FB_VN_ReadMlModel

VAR_INPUT
    sFilePath     : STRING;
    ipDestMlModel : Reference To ITcVnMlModel;
    bRead         : BOOL;
    nTimeout      : TIME;
END_VAR

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

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sFilePath</td>
<td>STRING</td>
<td>Full path of the file or relative path to the default directory on the target pc</td>
<td></td>
</tr>
<tr>
<td>ipDestMlModel</td>
<td>Reference To ITcVnMlModel</td>
<td></td>
<td>Returns the loaded machine learning model</td>
</tr>
<tr>
<td>bRead</td>
<td>BOOL</td>
<td></td>
<td>Reading the file is triggered by a rising edge at this input.</td>
</tr>
<tr>
<td>nTimeout</td>
<td>TIME</td>
<td>VISION_ADS_TIME_OUT</td>
<td>Indicates the time before the function is cancelled.</td>
</tr>
</tbody>
</table>
### Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bBusy</td>
<td>BOOL</td>
<td>This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.</td>
</tr>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrorId'. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.</td>
</tr>
<tr>
<td>nErrorId</td>
<td>UDINT</td>
<td>Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.</td>
</tr>
</tbody>
</table>

### Further information

The function block **FB_VN_ReadMlModel** loads an ML model from a file asynchronously into the PLC.

#### Loading of modified ML models.

When loading the ML models, the vision service caches the data internally. Loading modified ML models is therefore not possible in the standard system. To do this, adjust the cache settings or see the example Serial loading of images into the PLC for an alternative.

### Parameter

Some parameters are described centrally in the section [File Access Parameters](#).

**Loaded ML model (return value)**

The loaded **ITcVnMlModel** is reachable via the interface pointer **ipDestMlModel** following successful reading.

A special case is the use as **ITcVnColorModel**, because after loading the model a conversion with the following row is still necessary. Alternatively, the **ITcVnMlModel** can also be used directly with the corresponding functions.

```plaintext
hr := ipReadMlModel.TcQueryInterface(IID_ITcVnColorModel, ADR(ipColorModel));
```

### Application

For example, loading an ML model from the path **C:\TcVision\MlModel.vmlm** looks like this:

```plaintext
fbReadMlModel(
    sFilePath := 'C:\TcVision\MlModel.vmlm',
    ipDestMlModel := ipReadMlModel,
    bRead := TRUE,
    nTimeout := T#500MS
);
```

IF NOT fbReadMlModel.bBusy AND NOT fbReadMlModel.bError THEN
  // Ml-Model is accessible in ipReadMlModel
END_IF

### Required License

**TC3 Vision Base**
6.5.3.5  **FB_VN_WriteContainer**

This FB writes a container as an xml file to the hard drive.

**Syntax**

**Definition:**

```plaintext
FUNCTION_BLOCK FB_VN_WriteContainer

VAR_INPUT
    ipContainer : Reference To ITcVnContainer;
    sFilePath : STRING;
    eExportFormat : ETcVnContainerExportFormat;
    bWrite : BOOL;
    nTimeout : TIME;

VAR_OUTPUT
    bBusy : BOOL;
    bError : BOOL;
    nErrorId : UDINT;

END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContainer</td>
<td>Reference To ITcVnContainer [345]</td>
<td></td>
<td>The container to write</td>
</tr>
<tr>
<td>sFilePath</td>
<td>STRING</td>
<td></td>
<td>Full path of the file or relative path to the default directory on the target pc (e.g. only a file name). If the string is empty, a file name will be generated (containing the current time).</td>
</tr>
<tr>
<td>eExportFormat</td>
<td>ETcVnContainerExportFormat [175]</td>
<td>TCVN_CEF_XML</td>
<td>Container export format (default is human readable xml)</td>
</tr>
<tr>
<td>bWrite</td>
<td>BOOL</td>
<td></td>
<td>Writing the container is triggered by a rising edge at this input.</td>
</tr>
<tr>
<td>nTimeout</td>
<td>TIME</td>
<td>VISION_ADS_TIME OUT</td>
<td>Indicates the time before the function is cancelled.</td>
</tr>
</tbody>
</table>
## Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bBusy</td>
<td>BOOL</td>
<td>This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.</td>
</tr>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrorId'. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.</td>
</tr>
<tr>
<td>nErrorId</td>
<td>UDINT</td>
<td>Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.</td>
</tr>
</tbody>
</table>

### Further information

The function block **FB_VN_WriteContainer** saves a container [１43] from a file asynchronously into the PLC.

**Parameter**

Some parameters are described centrally in the section File Access Parameters [１299].

**Container**

The container **ipContainer** is saved as an XML file with the specified file path.

**Export format**

The export format **eExportFormat** specifies via the enum **EtcVnContainerExportFormat** [１75] whether the container is to be saved in a human-readable form or serialized:

- TCVN_CEF_XML
- TCVN_CEF_XML_SERIALIZED

The human-readable form has the advantage that entries in the container can be simply checked and edited manually. However, this can lead to rounding inaccuracies. The serialized format has the advantage that no rounding inaccuracies occur. However, the format is not easy to read or edit.

**Application**

The saving of a container in the serialized XML format in the file C:\TcVision\Container.xml looks like this, for example:

```plaintext
fbWriteContainer(
    ipContainer := ipContainer,
    sFilePath := 'C:\TcVision\Container.xml',
    eExportFormat := TCVN_CEF_XML_SERIALIZED,
    bWrite := TRUE,
    nTimeout := T#500MS
);
IF NOT fbWriteContainer.bBusy AND NOT fbWriteContainer.bError THEN
    // container was written successfully to file
END_IF
```

Related function block: **FB_VN_ReadContainer** [１302].

**Required License**

TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.5.3.6  **FB_VN_WriteImage**

This FB writes an image to the hard drive.

### Syntax

**Definition:**

```plaintext
FUNCTION_BLOCK FB_VN_WriteImage
    VAR_INPUT
        ipImage : Reference To ITcVnImage;
        sFilePath : STRING;
        bWrite : BOOL;
        nTimeout : TIME;
    END_VAR
    VAR_OUTPUT
        bBusy : BOOL;
        bError : BOOL;
        nErrorId : UDINT;
    END_VAR
END_FUNCTION_BLOCK
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage</td>
<td>Reference To ITcVnImage</td>
<td></td>
<td>The image to write</td>
</tr>
<tr>
<td>sFilePath</td>
<td>STRING</td>
<td></td>
<td>Full path of the file or relative path to the default directory on the target pc (e.g. only a file name). If the string is empty, a file name will be generated (containing the current time).</td>
</tr>
<tr>
<td>bWrite</td>
<td>BOOL</td>
<td></td>
<td>Writing the image is triggered by a rising edge at this input.</td>
</tr>
<tr>
<td>nTimeout</td>
<td>TIME</td>
<td>VISION_ADS_TIME_OUT</td>
<td>Indicates the time before the function is cancelled.</td>
</tr>
</tbody>
</table>
## Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bBusy</td>
<td>BOOL</td>
<td>This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.</td>
</tr>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrorId'. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.</td>
</tr>
<tr>
<td>nErrorId</td>
<td>UDINT</td>
<td>Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.</td>
</tr>
</tbody>
</table>

### Further information

The function block `FB_VN_WriteImage` saves images from a file asynchronously into the PLC.

### Parameter

Some parameters are described centrally in the section `File Access Parameters`.

### Input image

The format of the input image `ipImage` must match the file format defined in the file path `sFilePath`.

#### Saving on a rising edge of bWrite

The image is copied internally on a rising edge of the write command. Changes to the image in the following cycle thus have no effects. The image transferred at the time of the rising edge of bWrite is always the one that is saved.

### File path

The file path `sFilePath` specifies where the input image `ipImage` is saved and which image format should be used for saving. The image format is automatically recognized on the basis of the file extension. If no file name is specified it is automatically generated and contains the current time. If no file extension (e.g. `.jpg`) is specified, the extension `.bmp` and thus the BMP format are automatically used.

The following image formats are supported for saving.

- BMP (.bmp)
- PNG (.png)
- JPEG (.jpeg or .jpg)
- TIF (.tif)
- TwinCAT serialization (.tcimg)

#### NOTE

Possibility of errors

Make sure that the selected image format corresponds to the format of the actual input image. Otherwise, errors may occur or the saved images may be faulty.

#### NOTE

16-bit images

Only the TIF format and the TwinCAT serialization may be used if 16-bit images are to be saved without losses.
Application

The saving of an image in the default path for images looks like this, for example:

```plaintext
fbWriteImage(
    ipImage := ipImageRes,
    sFilePath := '','
    bWrite := TRUE,
    nTimeout := T#500MS
);

IF NOT fbWriteImage.bBusy AND NOT fbWriteImage.bError THEN
    // ipImageRes was successfully written to given file path.
END_IF
```

Examples

- Saving images from the PLC [1444]

Related function block: FB_VN_ReadImage [1304] for loading images.

Required License

TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.5.3.7 FB_VN_WriteMlModel

This FB writes a machine learning model to the hard drive.

Syntax

Definition:

```plaintext
FUNCTION_BLOCK FB_VN_WriteMlModel
VAR_INPUT
    ipMlModel : Reference To ITcVnMlModel;
    sFilePath : STRING;
    bWrite : BOOL;
    nTimeout : TIME;
END_VAR
VAR_OUTPUT
    bBusy : BOOL;
    bError : BOOL;
    nErrorId : UDINT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipMlModel</td>
<td>Reference To</td>
<td></td>
<td>The machine learning model to write</td>
</tr>
<tr>
<td></td>
<td>ITcVnMlModel</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[392]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sFilePath</td>
<td>STRING</td>
<td></td>
<td>Full path of the file or relative path to the default directory on the target pc (e.g. only a file name). If the string is empty, a file name will be generated (containing the current time).</td>
</tr>
<tr>
<td>bWrite</td>
<td>BOOL</td>
<td></td>
<td>Writing the machine learning model is triggered by a rising edge at this input.</td>
</tr>
<tr>
<td>nTimeout</td>
<td>TIME</td>
<td>VISION_ADS_TIME</td>
<td>Indicates the time before the function is cancelled.</td>
</tr>
</tbody>
</table>

## Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bBusy</td>
<td>BOOL</td>
<td>This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.</td>
</tr>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrorId'. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.</td>
</tr>
<tr>
<td>nErrorId</td>
<td>UDINT</td>
<td>Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.</td>
</tr>
</tbody>
</table>

## Further information

The function block FB_VN_WriteMlModel saves an ML model from the PLC asynchronously in a file.

### Parameter

Some parameters are described centrally in the section File Access Parameters [1299].

### ML model

The ML model `ipMlModel` is saved under the specified file path in binary format.

The saving of a `ITcVnColorModel` represents a special case, since this must still be converted first with the following row into a ML model. Alternatively, Color Model can be replaced by `ITcVnMlModel` with the corresponding functions.

```plaintext
hr := ipColorModel.TcQueryInterface(IID_ITcVnMlModel, ADR(ipWriteMlModel));
```

### Export format

If `sFilePath` is empty, a file name with the current time is generated and the extension vmlm is used. If you assign your own name, any ending is also possible.

### Application

For example, saving an ML model to the file `C:\TcVision\MlModel.vmlm` looks like this:
**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

**6.5.4 Image Acquisition**

This group contains function blocks for acquiring images with Vision devices (GigE Vision cameras and File Source objects).

**Function blocks**

- FB_VN_SimpleCameraControl for cameras and file sources
- FB_VN_GevCameraControl for GigE Vision cameras
- FB_VN_FileSourceControl for file sources

**General**

The function blocks in this group represent a proxy for the Image Provider TcCOM objects. Accordingly, they need not be called cyclically. Instead, the TcCOM object is controlled via the methods of the function block.

**State machine**

All function blocks for image acquisition include a state machine to manage the state of the represented device. Essentially, a function block must be placed in the ACQUIRING state in order to receive images continuously. The INITIAL state represents the starting point, after a restart or reset, from which the INITIALIZING can be triggered. From the OPENED state, the communication channel to the device is open in order, for example, to read or write register values.
All states are represented by the enum `ETcVnCameraState` and are explained below:

**Main states**

The main states are stationary states. They essentially indicate whether there is a connection to a Vision device (OPENED), whether the device is capturing images (ACQUIRING) and whether there is an error with the connection to the device (ERROR). You can switch between these states with the methods of the FBs. As a rule, the transition between two main states does not take place immediately, but over at least one cycle. In this intermediate time, the Vision device is in an intermediate state.

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR</td>
<td>Error state from which every device can only be reactivated by calling the method <code>Reset</code>.</td>
</tr>
<tr>
<td>INITIAL</td>
<td>Initial state of each device.</td>
</tr>
<tr>
<td>INITIALIZED</td>
<td>Similar to INITIAL, where the camera was manually initialized (e.g. with ForceIP and the initialization commands)</td>
</tr>
<tr>
<td>OPENED</td>
<td>There is a connection to the camera, individual images can be triggered and camera registers can be read and written. <strong>Note</strong>: This state can be skipped with the method <code>StartAcquisition</code>.</td>
</tr>
<tr>
<td>ACQUIRING</td>
<td>Device is in the acquisition state and is sending images depending on the settings.</td>
</tr>
</tbody>
</table>

**Intermediate states**

The intermediate states are necessary as most actions cannot be completed immediately (e.g. image acquisition). If the function blocks are in one of these intermediate states, the respective method must be called again in order to complete the transition to the respective main state.

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INITIALIZING</td>
<td>INITIAL &gt; INITIALIZED</td>
</tr>
<tr>
<td>OPENING</td>
<td>INITIAL &gt; OPENED and INITIALIZED &gt; OPENED</td>
</tr>
<tr>
<td>STARTACQUISITION</td>
<td>OPENED &gt; ACQUIRING</td>
</tr>
<tr>
<td>STOPACQUISITION</td>
<td>ACQUIRING &gt; OPENED</td>
</tr>
<tr>
<td>RESETTINGFEATURES</td>
<td>OPENED &gt; OPENED</td>
</tr>
<tr>
<td>TRIGGERING</td>
<td>OPENED &gt; OPENED and ACQUIRING &gt; ACQUIRING</td>
</tr>
<tr>
<td>CLOSING</td>
<td>OPENED &gt; INITIALIZED</td>
</tr>
</tbody>
</table>
Methods must be called in intermediate states!
A respective method for state transitions must also be called in the intermediate states. The state transition is only complete and the method no longer needs to be called when a stationary state is detected using the GetState method after a corresponding method call. Triggering a method once and subsequently waiting does not result in the desired state transition.

A state machine can be implemented in the PLC either with a CASE ... OF structure or with an IF ... ELSIF ... structure.

Samples
- Triggering an image by file name [1446]

6.5.4.1 FB_VN_FileSourceControl
This FB provides access to the images send by a File Source instance.
Do not call the main FB directly. Only use the available methods.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetCurrentImage [1319]</td>
<td>Gets the current available image (if any).</td>
</tr>
<tr>
<td>GetCurrentImageAndFileName</td>
<td>Gets the current available image (if any) and the corresponding file name.</td>
</tr>
<tr>
<td>StartAcquisition [1320]</td>
<td>Activates the File Source. If not in trigger mode, it will start sending images.</td>
</tr>
<tr>
<td>StopAcquisition [1321]</td>
<td>Deactivates the File Source. It will stop sending images.</td>
</tr>
<tr>
<td>Reset [1321]</td>
<td>Reset the controller to initial state (might require multiple calls depending on current state, until S_OK is returned)</td>
</tr>
<tr>
<td>GetState [1322]</td>
<td>Gets the current state of the internal camera control state machine.</td>
</tr>
<tr>
<td>TriggerImage [1322]</td>
<td>If in trigger mode, this function triggers the next image in the list.</td>
</tr>
<tr>
<td>TriggerImageExp [1323]</td>
<td>If in trigger mode, this function triggers a specific image in the list.</td>
</tr>
<tr>
<td>TriggerImageByName [1323]</td>
<td>If in trigger mode, this function triggers a specific image in the list by name.</td>
</tr>
</tbody>
</table>

Further information
The function block FB_VN_FileSourceControl can control File Source [119] instances with the full range of functions in the PLC. Alternatively, this is also possible with limited functionality with the function block FB_VN_SimpleCameraControl [1348].

State machine
The following diagram shows the state machine. The main states are marked in blue, the transition states are gray and the error state is red.
The following methods can be used to control the state machine:

- `GetState` [1322]
• **StartAcquisition** [► 1320]
• **StopAcquisition** [► 1321]
• **TriggerImage** [► 1322]
• **TriggerImageByName** [► 1323]
• **Reset** [► 1321]

If a method is called in a state for which it is not intended, the error code INVALIDSTATE (16#712) is returned.

**Application**

A state machine that continuously triggers an image with a special file name looks like this, for example:

```pascal
VAR
    fbFileSource : FB_VN_FileSourceControl;
    eState : TCVnCameraState;
    ipImageIn : TCVnImage;
    ipImageInDisp : TCVnDisplayableImage;
    bTrigger : BOOL := TRUE;
    sFileName : STRING := 'FileName.bmp';
    hr : HRESULT;
END_VAR

eState := fbFileSource.GetState();

CASE eState OF

    TCVN_CS_INITIAL, TCVN_CS_OPENING, TCVN_CS_OPENED, TCVN_CS_STARTACQUISITION:
        hr := fbFileSource.StartAcquisition();

    TCVN_CS_TRIGGERING:
        hr := fbFileSource.TriggerImage();

    TCVN_CS_ACQUIRING:
        IF bTrigger THEN
            hr := fbFileSource.TriggerImageByName(sFileName);
            IF SUCCEEDED(hr) THEN
                bTrigger := FALSE;
            END_IF
        ELSE
            hr := fbFileSource.GetCurrentImage(ipImageIn);
            IF SUCCEEDED(hr) AND ipImageIn <> 0 THEN
                bTrigger := TRUE;
                hr := F_VN_CopyIntoDisplayableImage(ipImageIn, ipImageInDisp, hr);
            END_IF
        END_IF

    TCVN_CS_ERROR:
        hr := fbFileSource.Reset();
END_CASE
```

**Samples**

• **Triggering an image by file name** [► 1446]

**Related function blocks**

• **FB_VN_SimpleCameraControl** [► 1348] for cameras and file sources
• **FB_VN_GevCameraControl** [► 1324] for GigE Vision cameras
• **FB_VN_FileSourceControl** [► 1316] for file sources

**Required License**

TC3 Vision Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g.</td>
<td>Tc3_Vision</td>
</tr>
<tr>
<td></td>
<td>Intel 4-core Atom CPU</td>
<td></td>
</tr>
</tbody>
</table>

6.5.4.1.1 GetCurrentImage

**GetCurrentImage**

```
ipImage Reference To ITcVnImage HRESULT GetCurrentImage```

Gets the current available image (if any).

**Syntax**

**Definition:**

```
METHOD GetCurrentImage : HRESULT
VAR_INPUT
  ipImage : Reference To ITcVnImage;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage</td>
<td>Reference To ITCvImage</td>
<td>Returns the image</td>
</tr>
</tbody>
</table>

**Return value**

`HRESULT [135]`

6.5.4.1.2 GetCurrentImageAndFileName

**GetCurrentImageAndFileName**

```
ipImage Reference To ITcVnImage sFileName STRING nMaxLen UINT HRESULT GetCurrentImageAndFileName```

Gets the current available image (if any) and the corresponding file name.

**Syntax**

**Definition:**

```
METHOD GetCurrentImageAndFileName : HRESULT
VAR_INPUT
  ipImage : Reference To ITcVnImage;
  sFileName : STRING;
  nMaxLen : UINT;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage</td>
<td>Reference To ITcVnImage</td>
<td>Returns the image</td>
</tr>
<tr>
<td>sFileName</td>
<td>STRING</td>
<td>Returns the file name</td>
</tr>
<tr>
<td>nMaxLen</td>
<td>UINT</td>
<td>Define the maximum allowed length for the file name</td>
</tr>
</tbody>
</table>

## Return value

HRRESULT [135]

### 6.5.4.1.3 StartAcquisition

StartAcquisition

Activates the File Source. If not in trigger mode, it will start sending images.

**Syntax**

**Definition:**

```plaintext
METHOD StartAcquisition : HRESULT
```

**Return value**

HRRESULT [135]

## Further information

The method StartAcquisition starts the loading of the images. Depending on the trigger mode setting, this may mean providing images directly or waiting for a trigger signal. The image loading can be started from the INITIAL and OPENED states.

### State transitions

With StartAcquisition the file source is to be placed in the ACQUIRING state. This can take place from the INITIAL and OPENED states. If the File Source is in INITIAL, the state transition to OPENED is performed first. As a rule, this takes place via the intermediate state OPENING. The transition from OPENED to ACQUIRING usually takes place via the intermediate state STARTACQUISITION.

All possible state transitions and their return codes are shown in the following transition matrix:

<table>
<thead>
<tr>
<th>Return Codes (HRESULT)</th>
<th>INITIAL</th>
<th>OPENING</th>
<th>OPENED</th>
<th>STARTACQUISITION</th>
<th>ACQUIRING</th>
<th>ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial state</td>
<td>INITIAL</td>
<td>BUSY</td>
<td>PENDING</td>
<td>SUCCESS</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OPENDING</td>
<td>OPENING</td>
<td>-</td>
<td>SUCCESS</td>
<td>-</td>
<td>-</td>
<td>FAILED</td>
</tr>
<tr>
<td>OPENED</td>
<td>OPENED</td>
<td>-</td>
<td>BUSY</td>
<td>PENDING</td>
<td>SUCCESS</td>
<td></td>
</tr>
<tr>
<td>STARTACQUISITION</td>
<td>STARTACQUISITION</td>
<td>-</td>
<td>BUSY</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
6.5.4.1.4  StopAcquisition

StopAcquisition

Deactivates the File Source. It will stop sending images.

Syntax

Definition:
METHOD StopAcquisition : HRESULT

Return value

HRESULT [135]

Further information

The method StopAcquisition stops the image loading of the File Source.

State transitions

With StopAcquisition, the File Source is to be brought from the ACQUIRING state to the OPENED state. As a rule, this doesn't happen immediately, but via the intermediate state STOPACQUISITION.

All possible state transitions and their return codes are shown in the following transition matrix:

<table>
<thead>
<tr>
<th>Return Codes (HRESULT)</th>
<th>Result state</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ACQUIRING</td>
</tr>
<tr>
<td>Initial state</td>
<td></td>
</tr>
<tr>
<td>ACQUIRING</td>
<td>BUSY</td>
</tr>
<tr>
<td>STOPACQUISITION</td>
<td>-</td>
</tr>
</tbody>
</table>

6.5.4.1.5  Reset

Reset

Reset the controller to initial state (might require multiple calls depending on current state, until S_OK is returned)

Syntax

Definition:
METHOD Reset : HRESULT

Return value

HRESULT [135]

Further information

The method Reset resets the File Source if it is in an error state.
State transitions

With Reset the File Source is to be brought from the ERROR state to the INITIAL state. If the reset is successful, this transition takes place immediately, otherwise the function block remains in the ERROR state.

All possible state transitions and their return codes are shown in the following transition matrix:

<table>
<thead>
<tr>
<th>Return Codes (HRESULT)</th>
<th>Result state</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial state</td>
<td>ERROR</td>
</tr>
<tr>
<td></td>
<td>FAILED</td>
</tr>
<tr>
<td></td>
<td>SUCCESS</td>
</tr>
</tbody>
</table>

In case of success, S_OK or S_FALSE (if there was no error to reset) is returned as SUCCESS code.

6.5.4.1.6 GetState

GetState

ETcVnCameraState GetState

Gets the current state of the internal camera control state machine.

Syntax

Definition:

METHOD GetState : ETcVnCameraState

Return value

ETcVnCameraState ▶ 158

Further information

The method GetState returns the state of the camera object or File Source object as a value of the enum ETcVnCameraState ▶ 158. The following states are possible:

• ERROR
• INITIAL
• OPENING
• OPENED
• STARTACQUISITION
• STOPACQUISITION
• ACQUIRING
• TRIGGERING

6.5.4.1.7 TriggerImage

TriggerImage

HRESULT TriggerImage

If in trigger mode, this function triggers the next image in the list.

Syntax

Definition:

METHOD TriggerImage : HRESULT
Return value

HRESULT [135]

Further information

The method TriggerImage triggers an image from the file source.

State transitions

With TriggerImage, the File Source is to load only one image while remaining in its current ACQUIRING state. As the image acquisition normally takes several cycles, however, the file source usually briefly enters the intermediate state TRIGGERING.

All possible state transitions and their return codes are shown in the following transition matrix:

<table>
<thead>
<tr>
<th>Return Codes (HRESULT)</th>
<th>Result state</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRIGGERING</td>
</tr>
<tr>
<td>Initial state</td>
<td>ACQUIRING</td>
</tr>
<tr>
<td>TRIGGERING</td>
<td>BUSY</td>
</tr>
</tbody>
</table>

6.5.4.1.8 TriggerImageExp

If in trigger mode, this function triggers a specific image in the list.

Syntax

Definition:

METHOD TriggerImageExp : HRESULT
VAR_INPUT
    nSkipImages : DINT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nSkipImages</td>
<td>DINT</td>
<td>Amount of images to skip, relative to current</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Further information

The state transitions of the method TriggerImageExp are the same as for TriggerImage [1322].

6.5.4.1.9 TriggerImageByName

If in trigger mode, this function triggers a specific image in the list by name.
Syntax

Definition:

METHOD TriggerImageByName : HRESULT
VAR_INPUT
   sImageName : STRING;
END_VAR

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sImageName</td>
<td>STRING</td>
<td>Name of the image to trigger</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

Further information

The state transitions of the method TriggerImageByName are the same as for TriggerImage [1322].

6.5.4.2 FB_VN_GevCameraControl

This FB provides the basic functionality to control a GigE Vision camera and access its calibration data. Do not call the main FB directly. Only use the available methods.
Methods
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClearImageQueue [1330]</td>
<td>Delete all images contained in the TcVnGevImageProvider TcCOM module receive queue and reset the corresponding omitted images counter.</td>
</tr>
<tr>
<td>CloseCamera [1330]</td>
<td>Close the control channel to the camera.</td>
</tr>
<tr>
<td>GetCalibPatternRef [1331]</td>
<td>Gets the reference calibration pattern points from the TcVnGevImageProvider TcCOM module (Can be set from the calibration assistant)</td>
</tr>
<tr>
<td>GetCameraMatrix [1332]</td>
<td>Gets the camera matrix from the TcVnGevImageProvider TcCOM module (requires intrinsic calibration parameters, e.g. set by the calibration assistant)</td>
</tr>
<tr>
<td>GetCurrentImage [1332]</td>
<td>Gets the current available image (first in receive queue).</td>
</tr>
<tr>
<td>GetCurrentImageUndistorted [1333]</td>
<td>Gets the current available image (first in receive queue) with undistortion applied (requires intrinsic calibration parameters in the TcVnGevImageProvider TcCOM module, e.g. set by the calibration assistant)</td>
</tr>
<tr>
<td>GetCurrentImageWithGvspInfo [1333]</td>
<td>Gets the current available image (first in receive queue) and its corresponding GVSP info.</td>
</tr>
<tr>
<td>GetCurrentImageWithGvspInfo Undistorted [1334]</td>
<td>Gets the current available image (first in receive queue) with its corresponding GVSP info and undistortion applied (requires intrinsic calibration parameters in the TcVnGevImageProvider TcCOM module, e.g. set by the calibration assistant)</td>
</tr>
<tr>
<td>GetDistortionCoefficients [1334]</td>
<td>Gets the distortion coefficients from the TcVnGevImageProvider TcCOM module (requires intrinsic calibration parameters, e.g. set by the calibration assistant)</td>
</tr>
<tr>
<td>GetLastImageFromQueue [1335]</td>
<td>Gets the last received image from the queue.</td>
</tr>
<tr>
<td>GetOmittedImagesNum [1335]</td>
<td>Gets the number of omitted images since the last call of ClearImageQueue (). If the image receive queue in the TcVnGevImageProvider TcCOM module is full and a new image arrives, the first one in the queue will be deleted and the omitted counter is increased.</td>
</tr>
<tr>
<td>GetRotationMatrix [1336]</td>
<td>Gets the rotation matrix from the TcVnGevImageProvider TcCOM module (requires extrinsic calibration parameters, e.g. set by the calibration assistant)</td>
</tr>
<tr>
<td>GetState [1336]</td>
<td>Gets the current state of the internal camera control state machine.</td>
</tr>
<tr>
<td>GetTranslationVector [1337]</td>
<td>Gets the translation vector from the TcVnGevImageProvider TcCOM module (requires extrinsic calibration parameters, e.g. set by the calibration assistant)</td>
</tr>
<tr>
<td>InitializeCamera [1337]</td>
<td>Initialize the camera to the intended state (includes sending the 'InitCameraCommands' defined in the GevImageAcquisition TcCOM module to the camera).</td>
</tr>
<tr>
<td>OpenCamera [1338]</td>
<td>Open a control channel to the camera.</td>
</tr>
<tr>
<td>Reset [1340]</td>
<td>Reset the camera controller to initial state (might require multiple calls depending on current state, until S_OK is returned)</td>
</tr>
<tr>
<td>ResetCameraFeatures [1340]</td>
<td>Reset the camera features to initial state (sends the 'InitCameraCommands' defined in the GevImageAcquisition TcCOM module to the camera).</td>
</tr>
<tr>
<td>SetCameraMatrix [1341]</td>
<td>Sets the camera matrix to the TcVnGevImageProvider TcCOM module</td>
</tr>
<tr>
<td>SetDistortionCoefficients [1342]</td>
<td>Sets the distortion coefficients to the TcVnGevImageProvider TcCOM module</td>
</tr>
<tr>
<td>SetRotationMatrix [1342]</td>
<td>Sets the rotation matrix to the TcVnGevImageProvider TcCOM module</td>
</tr>
<tr>
<td>SetTranslationVector [1343]</td>
<td>Sets the translation vector to the TcVnGevImageProvider TcCOM module</td>
</tr>
<tr>
<td>StartAcquisition [1343]</td>
<td>Send the 'StartAcquisitionCommands' defined in the TcVnGevImageProvider TcCOM module to the camera.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>StopAcquisition</td>
<td>Send the 'StopAcquisitionCommands' defined in the TcVnGevImageProvider TcCOM module to the camera.</td>
</tr>
<tr>
<td>TriggerImage</td>
<td>Send the 'SoftwareTriggerCommands' defined in the TcVnGevImageProvider TcCOM module to the camera.</td>
</tr>
</tbody>
</table>

**Further information**

The function block `FB_VN_GevCameraControl` represents GigE Vision Camera objects to the full extent in the PLC. Alternatively, the function block `FB_VN_SimpleCameraControl` can also be used for limited access to camera objects.

This function block is not intended to be called directly. In fact, the methods contain all functionalities.

**State machine**

The following diagram shows the state machine. The main states are marked in blue, the transition states are gray and the error state is red. The `GetState` method can be used to query the state of the camera. In addition, the methods with which a state transition can be achieved and the return value to which each leads are shown. If Init Commands are present and the Initialization Auto Mode AUTOINIT_AFTER_SO has been selected, the INITIALIZING state in which the automatic initialization takes place is also briefly assumed.
You can query the current state using the GetState method.

The following methods can be used to control the state machine:

- GetState

Fig. 26:
• InitializeCamera [1337]
• OpenCamera [1338]
• CloseCamera [1330]
• StartAcquisition [1343]
• StopAcquisition [1346]
• TriggerImage [1347]
• ResetCameraFeatures [1340]
• Reset [1340]

If a method is called in a state for which it is not intended, the error code INVALIDSTATE (16#712) is returned.

### HRESULT

<table>
<thead>
<tr>
<th>Hex</th>
<th>Dec</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>16#718</td>
<td>1816</td>
<td>NOTINIT</td>
<td>Camera has not yet been initialized. Initialize the camera either with the method InitializeCamera [1337] or by setting the TcCOM parameter InitializationAutoMode [94] to AUTOINIT_SO or AUTOINIT_AFTER_SO.</td>
</tr>
<tr>
<td>16#71A</td>
<td>1818</td>
<td>NOINTERFACE</td>
<td>The camera function block is not linked to the camera TcCOM object. Establish this link in the TwinCAT project instance under Symbol Initialization.</td>
</tr>
</tbody>
</table>

### Application

A state machine for placing a camera in image acquisition mode, handling error states and displaying successfully captured images looks like this, for example:

```plaintext
VAR
  hr : HRESULT;
  ipImage : ITcVnImage;
  ipImageOrgDisp : ITcVnDisplayableImage;
  fbCameraControl : FB_VN_GevCameraControl;
  eCameraState : ETCvNCameraState;
END_VAR

eCameraState := fbCameraControl.GetState();

// CameraControl is in error state, so try to reset the camera connection
IF eCameraState = TCVN_CS_ERROR THEN
  hr := fbCameraControl.Reset();

// Camera not yet initialized
ELSIF eCameraState < TCVN_CS_INITIALIZED THEN
  hr := fbCameraControl.InitializeCamera();

// Camera not yet opened
ELSIF eCameraState < TCVN_CS_OPENED THEN
  hr := fbCameraControl.OpenCamera();

// Camera not yet streaming
ELSIF eCameraState < TCVN_CS_ACQUIRING THEN
  hr := fbCameraControl.StartAcquisition();

// Camera streaming
ELSIF eCameraState = TCVN_CS_ACQUIRING THEN
  hr := fbCameraControl.GetCurrentImage(ipImage);
  IF SUCCEEDED(hr) AND ipImage <> 0 THEN
    // Do stuff...
    hr := F_VN_CopyIntoDisplayableImage(ipImage, ipImageOrgDisp, hr);
  END_IF
ENDIF
```

### Related function blocks

• FB_VN_SimpleCameraControl [1348] for cameras and file sources
API reference

• FB VN GevCameraControl [1324] for GigE Vision cameras
• FB VN FileSourceControl [1316] for file sources

Required License
TC3 Vision Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

6.5.4.2.1 ClearImageQueue

ClearImageQueue

HRESULT ClearImageQueue

Delete all images contained in the TcVnGevImageProvider TcCOM module receive queue and reset the corresponding omitted images counter.

Syntax

Definition:
METHOD ClearImageQueue : HRESULT

Return value

HRESULT [135]

6.5.4.2.2 CloseCamera

CloseCamera

HRESULT CloseCamera

Close the control channel to the camera.

Syntax

Definition:
METHOD CloseCamera : HRESULT

Return value

HRESULT [135]

Further information

The method CloseCamera closes an open communication channel to a camera.

State transitions

With CloseCamera, the camera is to be brought from the OPENED state to the INITIALIZED state. As a rule, this transition takes place via the intermediate state CLOSING.
Fig. 27: State transitions with FB_VN_GevCameraControl.CloseCamera

All possible state transitions and their return codes are shown in the following transition matrix:

<table>
<thead>
<tr>
<th>Return Codes (HRESULT)</th>
<th>Result state</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OPENED</td>
</tr>
<tr>
<td>Initial state</td>
<td>OPENED</td>
</tr>
<tr>
<td>CLOSING</td>
<td>-</td>
</tr>
</tbody>
</table>

### 6.5.4.2.3 GetCalibPatternRef

Gets the reference calibration pattern points from the TcVnGevImageProvider TcCOM module (Can be set from the calibration assistant)

**Syntax**

**Definition:**

```plaintext
METHOD GetCalibPatternRef : HRESULT
VAR_INPUT
   ipCalibPatternRef : Reference To ITcVnContainer;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipCalibPatternRef</td>
<td>Reference To ITcVnContainer</td>
<td>Returns the reference calibration pattern points (ContainerType_Vector_TcVnPoint3_REAL)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]
6.5.4.2.4 GetCameraMatrix

Gets the camera matrix from the TcVnGevImageProvider TcCOM module (requires intrinsic calibration parameters, e.g. set by the calibration assistant)

Syntax

Definition:

METHOD GetCameraMatrix : HRESULT
VAR_INPUT
   aCameraMatrix : Reference To TcVnMatrix3x3_LREAL;
END_VAR

 Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aCameraMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL</td>
<td>Returns the camera matrix</td>
</tr>
</tbody>
</table>

 Return value

HRESULT [135]

6.5.4.2.5 GetCurrentImage

Gets the current available image (first in receive queue).

Syntax

Definition:

METHOD GetCurrentImage : HRESULT
VAR_INPUT
   ipImage : Reference To ITcVnImage;
END_VAR

 Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage</td>
<td>Reference To ITcVnImage</td>
<td>Returns the image</td>
</tr>
</tbody>
</table>

 Return value

HRESULT [135]
6.5.4.2.6  GetCurrentImageUndistorted

**GetCurrentImageUndistorted**

- **ipImage**  
  Reference To ITCnVImage
- **HRESULT**  
  GetCurrentImageUndistorted

Gets the current available image (first in receive queue) with undistortion applied (requires intrinsic calibration parameters in the TcVnGevImageProvider TcCOM module, e.g. set by the calibration assistant).

### Syntax

**Definition:**

METHOD GetCurrentImageUndistorted : HRESULT
VAR_INPUT
  ipImage : Reference To ITCnVImage;
END_VAR

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage</td>
<td>Reference To ITCnVImage</td>
<td>Returns the undistorted image</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [ 135 ]

6.5.4.2.7  GetCurrentImageWithGvspInfo

**GetCurrentImageWithGvspInfo**

- **ipImage**  
  Reference To ITCnVImage
- **stGvspInfo**  
  Reference To GVSP_IMAGE_INFO
- **HRESULT**  
  GetCurrentImageWithGvspInfo

Gets the current available image (first in receive queue) and its corresponding GVSP info.

### Syntax

**Definition:**

METHOD GetCurrentImageWithGvspInfo : HRESULT
VAR_INPUT
  ipImage : Reference To ITCnVImage;
  stGvspInfo : Reference To GVSP_IMAGE_INFO;
END_VAR

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage</td>
<td>Reference To ITCnVImage</td>
<td>Returns the image</td>
</tr>
<tr>
<td>stGvspInfo</td>
<td>Reference To GVSP_IMAGE_INFO</td>
<td>Returns the GVSP info</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [ 135 ]
### 6.5.4.2.8 GetCurrentImageWithGvspInfoUndistorted

**GetCurrentImageWithGvspInfoUndistorted**

```
ipImage: Reference To ITcVnImage
stGvspInfo: Reference To GVSP_IMAGE_INFO
```

**HRESULT** GetCurrentImageWithGvspInfoUndistorted

Gets the current available image (first in receive queue) with its corresponding GVSP info and undistortion applied (requires intrinsic calibration parameters in the TcVnGevImageProvider TcCOM module, e.g. set by the calibration assistant).

**Syntax**

Definition:

```
METHOD GetCurrentImageWithGvspInfoUndistorted : HRESULT
VAR_INPUT
  ipImage : Reference To ITcVnImage;
  stGvspInfo : Reference To GVSP_IMAGE_INFO;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage</td>
<td>Reference To ITcVnImage [383]</td>
<td>Returns the undistorted image</td>
</tr>
<tr>
<td>stGvspInfo</td>
<td>Reference To GVSP_IMAGE_INFO [203]</td>
<td>Returns the GVSP info</td>
</tr>
</tbody>
</table>

**Return value**

**HRESULT [135]**

### 6.5.4.2.9 GetDistortionCoefficients

**GetDistortionCoefficients**

```
aDistortionCoefficients: Reference To TcVnArray8_LREAL
```

**HRESULT** GetDistortionCoefficients

Gets the distortion coefficients from the TcVnGevImageProvider TcCOM module (requires intrinsic calibration parameters, e.g. set by the calibration assistant).

**Syntax**

Definition:

```
METHOD GetDistortionCoefficients : HRESULT
VAR_INPUT
  aDistortionCoefficients : Reference To TcVnArray8_LREAL;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aDistortionCoefficients</td>
<td>Reference To TcVnArray8_LREAL [153]</td>
<td>Returns the distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]</td>
</tr>
</tbody>
</table>
Return value

HRESULT [135]

6.5.4.2.10 GetLastImageFromQueue

Syntax

Definition:

METHOD GetLastImageFromQueue : HRESULT
VAR_INPUT
  ipImage : Reference To ITCvImage;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage</td>
<td>Reference To ITCvImage</td>
<td>Returns the image</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.5.4.2.11 GetOmittedImagesNum

Syntax

Definition:

METHOD GetOmittedImagesNum : HRESULT
VAR_INPUT
  nOmitted : Reference To ULINT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOmitted</td>
<td>Reference To ULINT</td>
<td>Returns the number of omitted images</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]
6.5.4.2.12 GetRotationMatrix

Gets the rotation matrix from the TcVnGevImageProvider TcCOM module (requires extrinsic calibration parameters, e.g. set by the calibration assistant)

Syntax

Definition:

METHOD GetRotationMatrix : HRESULT
VAR_INPUT
  aRotationMatrix : Reference To TcVnMatrix3x3_LREAL;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aRotationMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL</td>
<td>Returns the rotation matrix</td>
</tr>
</tbody>
</table>

Return value

HRESULT [135]

6.5.4.2.13 GetState

Gets the current state of the internal camera control state machine.

Syntax

Definition:

METHOD GetState : ETcVnCameraState

Return value

ETcVnCameraState [158]

Further information

The method GetState returns the state of the camera object as a value of the enum ETcVnCameraState [158]. The following states are possible:

- ERROR
- INITIAL
- INITIALIZING
- INITIALIZED
- OPENING
- OPENED
6.5.4.2.14 GetTranslationVector

```
GetTranslationVector

aTranslationVector : Reference To TcVnVector3_LREAL

HRESULT GetTranslationVector
```

Gets the translation vector from the TcVnGevImageProvider TcCOM module (requires extrinsic calibration parameters, e.g. set by the calibration assistant)

**Syntax**

Definition:

```
METHOD GetTranslationVector : HRESULT
VAR_INPUT
    aTranslationVector : Reference To TcVnVector3_LREAL;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aTranslationVector</td>
<td>Reference To TcVnVector3_LREAL</td>
<td>Returns the translation vector</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

6.5.4.2.15 InitializeCamera

```
InitializeCamera

HRESULT InitializeCamera
```

Initialize the camera to the intended state (includes sending the 'InitCameraCommands' defined in the GevImageAcquisition TcCOM module to the camera).

**Syntax**

Definition:

```
METHOD InitializeCamera : HRESULT
```

**Return value**

HRESULT [135]
Further information

The method InitializeCamera initializes a connected GigE Vision camera just as the parameters in the stored initialization commands have been configured. This method should be used to explicitly control when the camera is initialized. To do this, the InitializationAutoMode in the TcCOM parameters of the Image Acquisition object [95] must be set to NO_AUTOINIT. As a result of this, no automatic camera initialization is performed; instead, it must be done manually by means of InitializeCamera.

If a different selection was made with InitializationAutoMode, an automatic initialization takes place and the InitializeCamera method is not required.

The initialization of a GigE Vision camera consists of the following steps:
1. Opening the Control Channel to the camera
2. Executing the Force-IP commands (optional)
3. Setting a UserSet or writing Initialization Commands [81].

The function block assumes the INITIALIZING state during initialization. The function block enters the INITIALIZED state once the initialization is complete. It also assumes this state if the camera connection in the OPENED state is closed by calling the method CloseCamera [1330].

State transitions

With InitializeCamera, the camera is to be brought from the INITIAL state to the INITIALIZED state. As a rule, this transition takes place via the intermediate state INITIALIZING.

![State transitions with FB_VN_GevCameraControl.InitializeCamera](image)

All possible state transitions and their return codes are shown in the following transition matrix:

<table>
<thead>
<tr>
<th>Return Codes (HRESULT)</th>
<th>INITIAL</th>
<th>INITIALIZING</th>
<th>INITIALIZED</th>
<th>ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Initial state</strong></td>
<td>INITIAL</td>
<td>BUSY</td>
<td>SUCCESS</td>
<td>FAILED</td>
</tr>
<tr>
<td></td>
<td>INITIALIZING</td>
<td>PENDING</td>
<td>BUSY</td>
<td></td>
</tr>
</tbody>
</table>

6.5.4.2.16 OpenCamera

OpenCamera

HRESULT OpenCamera
Open a control channel to the camera.

**Syntax**

**Definition:**

METHOD OpenCamera : HRESULT

**Return value**

HRESULT [135]

**Further information**

The method `OpenCamera` opens a communication channel to the camera. In the corresponding `OPENED` state, camera registers can be read and written, images can be acquired and continuous image recording can be started.

If the `InitializationAutoMode` is set to `AUTOINIT_AFTER_SO` in the TcCOM parameters of the Image Acquisition object [95], the GigE Vision camera is initialized the first time the method `OpenCamera` is called. Among others, this includes the writing of the Initialization Commands [81] or the setting of the UserSets. Alternatively, the `InitializeCamera` [1337] method can be called in order to perform the initialization manually.

**State transitions**

With `OpenCamera`, the camera is to be placed in the `OPENED` state. This can take place from the `INITIAL` and `INITIALIZED` states. As a rule, this transition takes place via the intermediate state `OPENING`.

![State transitions diagram](image)

Fig. 29: State transitions with FB_VN_GevCameraControl.OpenCamera

All possible state transitions and their return codes are shown in the following transition matrix:
### Return Codes (HRESULT)

<table>
<thead>
<tr>
<th>Initial state</th>
<th>INITIAL</th>
<th>INITIALIZED</th>
<th>OPENING</th>
<th>OPENED</th>
<th>ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td>INITIAL</td>
<td>BUSY</td>
<td>-</td>
<td>PENDING</td>
<td>SUCCESS</td>
<td>FAILED</td>
</tr>
<tr>
<td>INITIALIZED</td>
<td>-</td>
<td>BUSY</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPENING</td>
<td>-</td>
<td>-</td>
<td>BUSY</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### 6.5.4.2.17 Reset

Reset the camera controller to initial state (might require multiple calls depending on current state, until S_OK is returned)

**Syntax**

**Definition:**

METHOD Reset : HRESULT

- **Return value**
  
  HRESULT [135]

- **Further information**
  
  The method `Reset` resets the camera if it is in an error state.

- **State transitions**

  With `Reset` the camera function block should be brought from the ERROR state to the INITIAL state. If the reset is successful, this transition takes place immediately, otherwise the function block remains in the ERROR state.

  All possible state transitions and their return codes are shown in the following transition matrix:

<table>
<thead>
<tr>
<th>Return Codes (HRESULT)</th>
<th>Result state</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial state</td>
<td>ERROR</td>
</tr>
<tr>
<td>INITIAL</td>
<td>FAILED</td>
</tr>
</tbody>
</table>

  In case of success, S_OK or S_FALSE (if there was no error to reset) is returned as SUCCESS code.

#### 6.5.4.2.18 ResetCameraFeatures

Reset the camera features to initial state (sends the 'InitCameraCommands' defined in the GevlImageAcquisition TcCOM module to the camera)

**Syntax**

**Definition:**

METHOD ResetCameraFeatures : HRESULT
Return value

HRESULT [135]

Further information

The method ResetCameraFeatures resets the camera parameters to the original state. This means that the Initialization Commands [81], without the optional Force-IP command, will be rewritten to the camera.

State transitions

With ResetCameraFeatures the camera should only reset its parameters while remaining in the OPENED state. As a rule, however, the camera briefly enters the intermediate state RESETTNGFEATURES, as the parameter reset takes a little time.

Fig. 30: State transitions with FB_VN_GevCameraControl.ResetCameraFeatures

All possible state transitions and their return codes are shown in the following transition matrix:

<table>
<thead>
<tr>
<th>Return Codes (HRESULT)</th>
<th>Result state</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OPENED</td>
</tr>
<tr>
<td>Initial state</td>
<td>OPENED</td>
</tr>
<tr>
<td></td>
<td>RESSETINGFEATUR-ES</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6.5.4.2.19 SetCameraMatrix

SetCameraMatrix

Sets the camera matrix to the TcVnGevImageProvider TcCOM module

Syntax

Definition:

METHOD SetCameraMatrix : HRESULT
VAR_INPUT
  aCameraMatrix : Reference To TcVnMatrix3x3_LREAL;
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aCameraMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL</td>
<td>The camera matrix to be copied to the TcCOM module</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

### 6.5.4.2.20 SetDistortionCoefficients

Sets the distortion coefficients to the TcVnGevImageProvider TcCOM module

#### Syntax

**Definition:**

METHOD SetDistortionCoefficients : HRESULT  
VAR_INPUT  
   aDistortionCoefficients : Reference To TcVnArray8_LREAL;  
END_VAR

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aDistortionCoefficients</td>
<td>Reference To TcVnArray8_LREAL [153]</td>
<td>The distortion coefficients to be copied to the TcCOM module</td>
</tr>
</tbody>
</table>

## Return value

HRESULT [135]

### 6.5.4.2.21 SetRotationMatrix

Sets the rotation matrix to the TcVnGevImageProvider TcCOM module

#### Syntax

**Definition:**

METHOD SetRotationMatrix : HRESULT  
VAR_INPUT  
   aRotationMatrix : Reference To TcVnMatrix3x3_LREAL;  
END_VAR
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aRotationMatrix</td>
<td>Reference To TcVnMatrix3x3_LREAL</td>
<td>The rotation matrix to be copied to the TcCOM module</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

---

#### 6.5.4.2.22 SetTranslationVector

Sets the translation vector to the TcVnGevImageProvider TcCOM module

**Syntax**

**Definition:**

METHOD SetTranslationVector : HRESULT  
VAR_INPUT  
    aTranslationVector : Reference To TcVnVector3_LREAL;  
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aTranslationVector</td>
<td>Reference To TcVnVector3_LREAL</td>
<td>The translation vector to be copied to the TcCOM module</td>
</tr>
</tbody>
</table>

### Return value

HRESULT [135]

---

#### 6.5.4.2.23 StartAcquisition

Send the 'StartAcquisitionCommands' defined in the TcVnGevImageProvider TcCOM module to the camera.

**Syntax**

**Definition:**

METHOD StartAcquisition : HRESULT

**Return value**

HRESULT [135]
Further information

The method `StartAcquisition` starts the image acquisition by a camera. What this means in the individual case depends on the parameterization of the camera. Further information about this can be found in the sample `Image acquisition and trigger` [1459]. The image acquisition can be started from the `INITIAL`, `INITIALIZED` and `OPENED` states.

If the camera is still in `INITIAL` or `INITIALIZED`, the method `StartAcquisition` also performs the tasks of the method `OpenCamera` [1338]. Usually, the `OPENED` state is not assumed.

State transitions

With `StartAcquisition` the camera is to be placed in the `ACQUIRING` state. This can take place from the `INITIAL`, `INITIALIZED` and `OPENED` states. If the camera is in `INITIAL` or `INITIALIZED`, the state transition to `OPENED` is performed first. As a rule, this takes place via the intermediate state `OPENING`. The transition from `OPENED` to `ACQUIRING` usually takes place via the intermediate state `STARTACQUISITION`. 
Fig. 31: State transitions with FB_VN_GevCameraControl.StartAcquisition

All possible state transitions and their return codes are shown in the following transition matrix:
### 6.5.4.2.24 StopAcquisition

Send the ‘StopAcquisitionCommands’ defined in the TcVnGevImageProvider TcCOM module to the camera.

**Syntax**

**Definition:**

```c
METHOD StopAcquisition : HRESULT
```

- **Return value**

  **HRESULT [135]**

- **Further information**

  The method `StopAcquisition` stops the image acquisition by the camera.

- **State transitions**

  With `StopAcquisition`, the camera is to be brought from the `ACQUIRING` state to the `OPENED` state. As a rule, this doesn’t happen immediately, but via the intermediate state `STOPACQUISITION`. 

---

<table>
<thead>
<tr>
<th><strong>Return Codes (HRESULT)</strong></th>
<th><strong>Result state</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>INITIAL</strong></td>
<td><strong>INITIALIZED</strong></td>
</tr>
<tr>
<td>INITIAL</td>
<td>BUSY</td>
</tr>
<tr>
<td>INITIALIZED</td>
<td>-</td>
</tr>
<tr>
<td>OPENING</td>
<td>-</td>
</tr>
<tr>
<td>OPENED</td>
<td>-</td>
</tr>
<tr>
<td>STARTACQUISITION</td>
<td>-</td>
</tr>
</tbody>
</table>
All possible state transitions and their return codes are shown in the following transition matrix:

<table>
<thead>
<tr>
<th>Return Codes (HRESULT)</th>
<th>Result state</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ACQUIRING</td>
<td>STOPACQUISITION</td>
</tr>
<tr>
<td>Initial state</td>
<td>ACQUIRING BUSY</td>
<td>PENDING SUCCESS</td>
</tr>
<tr>
<td>STOPACQUISITION</td>
<td>-</td>
<td>BUSY</td>
</tr>
</tbody>
</table>

### 6.5.4.2.25 TriggerImage

**Definition**

METHOD TriggerImage : HRESULT

**Syntax**

Send the 'SoftwareTriggerCommands' defined in the TcVnGevImageProvider TcCOM module to the camera.

**Return value**

HRESULT [135]

**Further information**

Configuring the camera: Image acquisition and trigger [1459]

**State transitions**

With TriggerImage the camera is to acquire only one image while remaining in the current ACQUIRING or OPENED state. As the image acquisition normally takes several cycles, the camera usually briefly enters the intermediate state TRIGGERING.
All possible state transitions and their return codes are shown in the following transition matrix:

<table>
<thead>
<tr>
<th>Return Codes (HRESULT)</th>
<th>Opened</th>
<th>Acquiring</th>
<th>Triggering</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial state</td>
<td>Opened</td>
<td>Busy</td>
<td>PENDING</td>
<td>FAILED</td>
</tr>
<tr>
<td></td>
<td>Acquiring</td>
<td>BUSY</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Triggering</td>
<td>SUCCESS</td>
<td>SUCCESS</td>
<td>BUSY</td>
</tr>
</tbody>
</table>

6.5.4.3 **FB_VN_SimpleCameraControl**

This FB provides the basic functionality to control a camera or a FileSource. Do not call the main FB directly. Only use the available methods.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetCurrentImage</td>
<td>Get the current available image (if any).</td>
</tr>
<tr>
<td>GetState</td>
<td>Get the current state of the internal camera control state machine.</td>
</tr>
<tr>
<td>Reset</td>
<td>Reset the camera to initial state (might require multiple calls depending on current state, until S_OK is returned)</td>
</tr>
<tr>
<td>StartAcquisition</td>
<td>Start the image acquisition.</td>
</tr>
<tr>
<td>StopAcquisition</td>
<td>Stop the image acquisition.</td>
</tr>
<tr>
<td>TriggerImage</td>
<td>Trigger the next image.</td>
</tr>
</tbody>
</table>

Further information

The function block FB_VN_SimpleCameraControl combines the commonalities of FB_VN_GevCameraControl and FB_VN_FileSourceControl. This means that instances of the function block can be linked to both the Image Provider of a camera and the Image Provider of a File Source Control, as in First Steps.

State machine

The following diagram shows the state machine. The main states are marked in blue, the transition states are gray and the error state is red. In contrast to a File Source Control, a camera always also assumes the transition state TCVN_CS_OPENING during the transition from TCVN_CS_INITIAL to TCVN_CS_ACQUIRING and, if necessary, also the main state TCVN_CS_OPENED. If Init Commands are present and the Initialization Auto Mode AUTOINIT_AFTER_SO has been selected, the INITIALIZING state in which the automatic...
Initialization takes place is also briefly assumed. If a transition occurs very quickly, the corresponding state is not reported to the outside, because the subsequent state has already been reached within a cycle. When linked to a File Source Control, these three states are not assumed. The use of the Trigger Image method from the \texttt{TCVN\_CS\_OPENED} state is only possible with one camera.

The following methods can be used to control the state machine:
• GetState [1351]
• StartAcquisition [1352]
• StopAcquisition [1353]
• TriggerImage [1353]
• Reset [1352]

If a method is called in a state for which it is not intended, the error code INVALIDSTATE (16#712) is returned.

**Application**

```plaintext
PROGRAM MAIN
VAR
    hr : HRESULT;
    ipImageIn : ITcVnImage;
    ipImageInDisp : ITcVnDisplayableImage;
    fbCameraControl : FB_VN_SimpleCameraControl;
    eCameraState : ETcVnCameraState;
END_VAR

eCameraState := fbCameraControl.GetState();

// CameraControl is in error state, so try to reset the camera connection
IF eCameraState = TCVN_CS_ERROR THEN
    hr := fbCameraControl.Reset();

// Camera not yet streaming
ELSIF eCameraState < TCVN_CS_ACQUIRING THEN
    hr := fbCameraControl.StartAcquisition();

// Camera streaming
ELSIF eCameraState = TCVN_CS_ACQUIRING THEN
    hr := fbCameraControl.GetCurrentImage(ipImageIn);
    IF SUCCEEDED(hr) AND ipImageIn <> 0 THEN
        // Do stuff...
        hr := F_VN_CopyIntoDisplayableImage(ipImageIn, ipImageInDisp, hr);
    END_IF
END_IF
```

**Samples**

In many function samples [1363], a FB_VN_SimpleCameraControl function block is used to load sample images.

**Related functions**

- FB_VN_SimpleCameraControl [1348] for cameras and file sources
- FB_VN_GevCameraControl [1324] for GigE Vision cameras
- FB_VN_FileSourceControl [1316] for file sources

**Required License**

TC3 Vision Base

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>
### 6.5.4.3.1 GetCurrentImage

**GetCurrentImage**

\[\text{ipImage : Reference To ITcVnImage, HRESULT : GetCurrentImage}\]

Get the current available image (if any).

**Syntax**

Definition:

```
METHOD GetCurrentImage : HRESULT
VAR_INPUT
  ipImage : Reference To ITcVnImage;
END_VAR
```

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage</td>
<td>Reference To ITcVnImage</td>
<td>Returns the image</td>
</tr>
</tbody>
</table>

#### Return value

HRESULT [135]

### 6.5.4.3.2 GetState

**GetState**

\[ETcVnCameraState, GetState\]

Get the current state of the internal camera control state machine.

**Syntax**

Definition:

```
METHOD GetState : ETcVnCameraState
```

#### Return value

ETcVnCameraState [158]

**Further information**

The method GetState returns the state of the camera object or File Source object as a value of the enum ETcVnCameraState [158]. The following states are possible:

- ERROR
- INITIAL
- OPENING
- OPENED
- STARTACQUISITION
- STOPACQUISITION
- ACQUIRING
- TRIGGERING
6.5.4.3.3 Reset

Reset

Reset the camera to initial state (might require multiple calls depending on current state, until S_OK is returned)

Syntax

Definition:

METHOD Reset : HRESULT

Return value

HRESULT [135]

Further information

The method Reset resets the device if it is in an error state.

State transitions

With Reset the device is to be brought from the ERROR state to the INITIAL state. If the reset is successful, this transition takes place immediately, otherwise the function block remains in the ERROR state.

All possible state transitions and their return codes are shown in the following transition matrix:

<table>
<thead>
<tr>
<th>Return Codes (HRESULT)</th>
<th>Result state</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR</td>
<td>FAILED</td>
</tr>
<tr>
<td>INITIAL</td>
<td>SUCCESS</td>
</tr>
</tbody>
</table>

In case of success, S_OK or S_FALSE (if there was no error to reset) is returned as SUCCESS code.

6.5.4.3.4 StartAcquisition

StartAcquisition

Start the image acquisition.

Syntax

Definition:

METHOD StartAcquisition : HRESULT

Return value

HRESULT [135]

Further information

• In connection with a camera, this method corresponds to the StartAcquisition [1343] method of FB_VN_GevCameraControl [1324]
• In connection with a File Source Control this method corresponds to the StartAcquisition [1320] method of FB_VN_FileSourceControl [1316]
6.5.4.3.5 StopAcquisition

StopAcquisition

Stop the image acquisition.

Syntax

Definition:

METHOD StopAcquisition : HRESULT

Return value

HRESULT [135]

Further information

- In connection with a camera, this method corresponds to the StopAcquisition [1346] method of FB_VN_GevCameraControl [1324].
- In connection with a File Source Control this method corresponds to the StopAcquisition [1321] method of FB_VN_FileSourceControl [1316].

6.5.4.3.6 TriggerImage

TriggerImage

Trigger the next image.

Syntax

Definition:

METHOD TriggerImage : HRESULT

Return value

HRESULT [135]

Further information

- In connection with a camera, this method corresponds to the TriggerImage [1347] method of FB_VN_GevCameraControl [1324].
- In connection with a File Source Control this method corresponds to the TriggerImage [1347] method of FB_VN_FileSourceControl [1316].

6.5.5 Image Processing

This group contains function blocks for image processing. In these complex function blocks, the values once set are retained after an execution. Furthermore, the function blocks can also contain several methods.

The functions contained in the chapter Functions [401], on the other hand, have no internal status information and the variable values are not retained.
Function blocks

- `FB_VN_GeneralizedHoughBallard` [1354]

### 6.5.5.1 FB_VN_GeneralizedHoughBallard

This FB provides the Generalized Hough Ballard functionality. First, set the parameters via the corresponding methods (optional, defaults are used otherwise). Then, set the template (required). Afterwards, the Detect method can be executed.

Do not call the main FB directly. Only use the available methods.

#### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Detect</code> [1355]</td>
<td>Detect the template in an image (SetTemplate must have been called before).</td>
</tr>
<tr>
<td><code>SetCannyThreshold</code> [1356]</td>
<td>Sets the canny edge detection thresholds.</td>
</tr>
<tr>
<td><code>SetInvAccuRatio</code> [1356]</td>
<td>Sets the inverted ratio of the accumulator size in relation to the source image's size (e.g. a value of 2 means that the size is halved in both directions.)</td>
</tr>
<tr>
<td><code>SetLevels</code> [1357]</td>
<td>Sets the number of R-table levels.</td>
</tr>
<tr>
<td><code>SetMinDist</code> [1357]</td>
<td>Sets the minimum distance between the centers of different objects.</td>
</tr>
<tr>
<td><code>SetTemplate</code> [1357]</td>
<td>Sets the template image to search for in the Detect method.</td>
</tr>
<tr>
<td><code>SetVotesThreshold</code> [1358]</td>
<td>Sets the accumulator threshold, i.e. the number of votes required to detect a match (too small values lead to false detections).</td>
</tr>
</tbody>
</table>

Further information

The function block `FB_VN_GeneralizedHoughBallard` finds an arbitrary shape in a grayscale image using the generalized Hough transformation. The center positions of the recognized template shapes are determined, whereby different scaling and rotation of the template are not taken into account.

Parameter

**Input images**

The template and input image must be a 1-channel image with element type USINT (8-bit). First, the template image must be passed using the `SetTemplate` method. After that, the execution can be done by calling the `Detect` method, where the input image is then passed.

**Parameters for detection**

The following parameters can be set by method call. Otherwise, these default values are used. Since the parameters are already used for internal calculations when the template image is passed, the values must always be set first.

- CannyThresholdLow = 50
- CannyThresholdHigh = 100
- InvAccuRatio = 1.0
- Levels = 360
- MinDist = 1.0
- VotesThreshold = 100

List of center points found (return value)
All center point positions found in the image are returned after calling the **Detect** method in the container `ipPositions`.

**Application**

The execution consists of 3 steps. First the parameters are set and then the template image is passed. After that it is sufficient to call only the **Detect** method with the images on which to search for the template. Only if parameters or the template are to be changed, the corresponding steps have to be executed again.

```c
// Set parameters first
hr := fbGenHoughBallard.SetCannyThreshold(50,150);
IF SUCCEEDED(hr) THEN hr := fbGenHoughBallard.SetInvAccuRatio(2); END_IF
IF SUCCEEDED(hr) THEN hr := fbGenHoughBallard.SetLevels(180); END_IF
IF SUCCEEDED(hr) THEN hr := fbGenHoughBallard.SetMinDist(20); END_IF
IF SUCCEEDED(hr) THEN hr := fbGenHoughBallard.SetVotesThreshold(80); END_IF

// Check if successful and set template image
IF SUCCEEDED(hr) THEN hr := fbGenHoughBallard.SetTemplate(ipTemplateImage); END_IF

// Check if successful and call detection
IF SUCCEEDED(hr) THEN hr := fbGenHoughBallard.Detect(ipSrcImage,ipPositions); END_IF
```

**Required License**

TC3 Vision Matching

**System Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.17 or later</td>
<td>PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU</td>
<td>Tc3_Vision</td>
</tr>
</tbody>
</table>

### 6.5.5.1.1 Detect

Detect the template in an image (SetTemplate must have been called before).

**Syntax**

**Definition:**

```c
METHOD Detect : HRESULT
VAR_INPUT
    ipImage : ITcVnImage;
    ipPositions : Reference To ITcVnContainer;
END_VAR

HRESULT Detect(ipImage, ipPositions)
```

Detect the template in an image (SetTemplate must have been called before).

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage</td>
<td><a href="#">ITcVnImage</a></td>
<td>Source image (1 channel, USINT)</td>
</tr>
<tr>
<td>ipPositions</td>
<td>Reference To ITcVnContainer [*]</td>
<td>Returns the centers of the detected template positions (ContainerType_Vector_TcVnPoint2_REAL; Non-zero interface pointers are reused.)</td>
</tr>
</tbody>
</table>

**Return value**

**HRESULT** [*]
6.5.5.1.2 SetCannyThreshold

Sets the canny edge detection thresholds.

Syntax

Definition:

METHOD SetCannyThreshold : HRESULT
VAR_INPUT
  nLow : DINT;
  nHigh : DINT;
END_VAR

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nLow</td>
<td>DINT</td>
<td>Low threshold (&gt; 0)</td>
</tr>
<tr>
<td>nHigh</td>
<td>DINT</td>
<td>High threshold (&gt; nLow, usually 2 to 3 * nLow)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Return value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HRESULT [135]</td>
</tr>
</tbody>
</table>

6.5.5.1.3 SetInvAccuRatio

Sets the inverted ratio of the accumulator size in relation to the source image's size (e.g. a value of 2 means that the size is halved in both directions.)

Syntax

Definition:

METHOD SetInvAccuRatio : HRESULT
VAR_INPUT
  fInvAccuRatio : LREAL;
END_VAR

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fInvAccuRatio</td>
<td>LREAL</td>
<td>Inverted accumulator ratio (&gt; 0)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Return value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HRESULT [135]</td>
</tr>
</tbody>
</table>

6.5.5.1.4  SetLevels

**SetLevels**

```c
nLevels  DINT  HRESULT  SetLevels
```

Sets the number of R-table levels.

**Syntax**

**Definition:**

```
METHOD SetLevels : HRESULT
VAR_INPUT
  nLevels : DINT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nLevels</td>
<td>DINT</td>
<td>Number of R-table levels (&gt; 0)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

6.5.5.1.5  SetMinDist

**SetMinDist**

```c
fMinDist  LREAL  HRESULT  SetMinDist
```

Sets the minimum distance between the centers of different objects.

**Syntax**

**Definition:**

```
METHOD SetMinDist : HRESULT
VAR_INPUT
  fMinDist : LREAL;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fMinDist</td>
<td>LREAL</td>
<td>Minimum distance (&gt; 0)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

6.5.5.1.6  SetTemplate

**SetTemplate**

```c
ipImage  ITCVnImage  HRESULT  SetTemplate
```

TF7000 - TF7300  Version: 1.3  1357
Sets the template image to search for in the Detect method.

**Syntax**

**Definition:**

METHOD SetTemplate : HRESULT
VAR_INPUT
  ipImage : ITcVnImage;
END_VAR

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipImage</td>
<td>ITcVnImage</td>
<td>Template image (1 channel, USINT)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]

**6.5.5.1.7 SetVotesThreshold**

**SetVotesThreshold**

Sets the accumulator threshold, i.e. the number of votes required to detect a match (too small values lead to false detections).

**Syntax**

**Definition:**

METHOD SetVotesThreshold : HRESULT
VAR_INPUT
  nVotes : DINT;
END_VAR

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nVotes</td>
<td>DINT</td>
<td>Number of votes required to detect a match (&gt; 0)</td>
</tr>
</tbody>
</table>

**Return value**

HRESULT [135]
This chapter deals with the interaction of TwinCAT Vision and the TwinCAT HMI. At present this is essentially a server extension that ensures that images from the PLC can be displayed in the TwinCAT HMI.

### Integration of packages

The integration of packages takes place in the case of TwinCAT HMI versions from 1.12 via the package management NuGet. In a TwinCAT HMI project, right-click References and select Manage NuGet Packages…. Then click Browse and select "TwinCAT HMI Official" as the Package source. You will now see a list of the officially available TwinCAT HMI packages from which you can select Beckhoff.TwinCAT.HMI.Vision.win-x64. Refer to the TwinCAT HMI documentation for detailed information.

### 7.1 Server extension

The TwinCAT HMI NuGet package Beckhoff.TwinCAT.HMI.Vision contains the HMI server extension TcHmiVision, with which images from the PLC can be displayed in the TwinCAT HMI. For this purpose, all ADS symbols of the type TcVnDisplayableImage from the configured runtimes are listed and can be linked in various formats with an HMI symbol.

### Configuration

Once the package has been successfully integrated into the HMI project, enter the target PLC under Server > Extensions > TcHmiVision:
Application

1. Navigate to an HMI view and add an Image control.

2. Open the properties of the control element and click on Create data binding... under the parameter Src...

3. Link the Src parameter of the image control with the corresponding icon as follows: Server symbols > TcHmiVision > PLC > MAIN > iplImage > < image format>
**PLC process image on the target system**
This is only possible if there is an appropriate process image of the PLC on the target system.

**Update interval**
The speed with which the images can be retrieved from the PLC is influenced by the system, the images themselves and the utilization rate. We therefore recommend specifying a fixed update interval in the binding dialog under **Settings**.

**Transmission formats**
The server extension contains the image data from the PLC in BMP format and forwards them as a Base64 string to the TwinCAT HMI client. In order to reduce the data quantity transmitted to a client, images can be compressed in the server extension. The following image formats are available:

- **BMP**: uncompressed
- **JPEG**: compressed (lossless), good for natural images
- **PNG**: compressed (lossless), good for artificially created images
- **Stream**: For continuous image streams (e.g. live camera picture, not recommended!)

**Note** For transmission as a stream the HMI server must be started as administrator. Otherwise no image transmission is possible.
8 **Samples**

In this chapter you will find samples for the application of many functions and function blocks in the PLC, as well as wizards and camera configuration. Before you start looking at the PLC samples, please have a look at First steps [► 25].

The samples are independent of each other. Each sample has a focus under which it is grouped below. Samples that focus on a function or a function block are listed based on the API structure.

**Download**

The samples can be obtained from a repository on GitHub: [https://github.com/Beckhoff/TF7xxx_Samples](https://github.com/Beckhoff/TF7xxx_Samples). There you have the option to clone the repository or download a ZIP file containing all samples.

**Sample images must be downloaded separately**

The GitHub repository contains only the TwinCAT projects with the corresponding PLC code. The image data "TF7xxx_Sample_Images.zip" must be downloaded separately from the Beckhoff website. The link and more information can also be found in the Readme file in the GitHub repository.

**Project resources**

Each sample is provided in a separate folder, according to the outline order and with the name of the chapter headings. In it there is a TwinCAT project with the corresponding PLC code.

The separate zip folder containing the sample images, for example, can be unzipped to the Images folder within the repository. The folder names as well as the structure in which the sample images are stored is identical to the project structure. Therefore, the assignment of the images is simply done via the path with the same folder name.

Before you execute a project, please adjust the system configuration [► 29] according to your target system, with particular attention to configuration of the cores, the router memory and the PLC cycle time. Furthermore, check the version of the TcCOM objects of camera or file source as well as the PLC library as described in the Version overview [► 15] and adjust the version if necessary.

In the GitHub repository you can find a PowerShell script (SetTcCOMModuleVersion.ps1) which adjusts the version in all projects of the PLC folder. To do this, run the script, e.g. via the PowerShell console, specifying the desired version as follows:

```
PS C:\TF7xxx_Samples> .\SetTcCOMModuleVersion.ps1 4.0.2.13
```
If images or streams are needed, integrate them via the File Source Control [120] or the Record/Playback [97] tab of the camera object.

8.1 Function samples

All of the following samples focus on a single function and are sorted in accordance with the TwinCAT Vision API reference. For each sample there are corresponding images that can be loaded to the PLC via the File Source. In addition, the respective sample descriptions show which function parameters have to be changed in order to see certain effects.

The general steps for starting the function samples are:

1. Add the attached sample images to the File Source Control.
   
   **Note:** When using your own images, please pay attention to the expected image format, which may vary depending on the sample.

2. Activate the configuration and switch TwinCAT and the PLC to Run mode.

   You can now see the images from the PLC via View > Other Windows > ADS Image Watch. Then
   - ipImageInDisp always displays the input image and
   - ipImageResDisp always displays the result image
   (Both image names can of course be different in your case, we have always named the variables the same in the examples, so that they can be easily recognized and you can quickly find your way around each project.)
   - If required, other images show results of intermediate steps or further information.
   Details can be found in the respective sample description.

In the sample descriptions you will also find the function parameters to be written in logged-in Run mode via 

**Prepared value**, followed by click on **write values**. The effects can then be seen again in the images displayed in Ads Image Watch.

Samples

- **Basic Image Operations [1366]**
- **Code Reading [1369]**
- **Contour analysis [1380]**
- **Image analysis [1390]**
- **Image Color and Contrast Processing [1410]**
- **Image segmentation [1420]**
- **Measurement [1423]**

Sample of writing your own functions

If you repeatedly use certain sequences of TwinCAT Vision API functions for your application, it is a good idea to encapsulate them in a function. The following sample explains what you should observe when doing this:

**Sample: Self-written functions [1440]**
8.1.1 Basic Container Operations

8.1.1.1 Selections of container elements

In this sample, container elements are filtered with the help of a user-defined condition. The function `F_VN_CopyContainerElementsConditional_ITcVnContainer` is used for this, as the container elements are in turn containers. If the container elements are iterators, the function `F_VN_CopyContainerElementsConditional_ITcVnForwardIterator` can also be used for this.

Explanation

If the elements of multidimensional containers are to be filtered, this can quickly result in a great deal of programming effort due to the access to the sub-elements. To counteract this, TwinCAT Vision provides a function precisely for this, which is used as follows, for example:

A function block is created that implements the interface `ITcVnCustomElementCondition_ITcVnContainer`. This interface possesses the `Condition` method, which accepts an element of the type `ITcVnContainer`. The filter condition that is to be applied to the element is defined in the implementation of this method. The function block with the filter condition is then transferred to the function `F_VN_CopyContainerElementsConditional_ITcVnContainer` in order to carry out the filtering of all elements of a container. In this sample the elements are filtered according to whether they possess more than five sub-elements.

Application

By way of example, the sub-containers of a container are filtered according to how many elements each of them possesses. Specifically, only those sub-containers that have more than five elements are to be accepted into a new container. The structure of the container can be varied by means of the array `aContainerStructure`. The elements indicate how many sub-elements each sub-container has. If the number of sub-containers itself is likewise varied, the constant `cNumberOfSubContainers` must be matched to it.

Program

MAIN

First of all a constant integer is defined in order to be able to simply vary the sub-containers to be filtered:

```plaintext
VAR CONSTANT
cNumberOfSubContainers : INT := 5;
END_VAR
```

Then all necessary variables are declared:

```plaintext
VAR
aContainerStructure : ARRAY [0..(cNumberOfSubContainers-1)] OF INT := [2, 7, 3, 5, 12];
ipHelper : ITcVnContainer;
ipContainerBase : ITcVnContainer;
ipContainerFiltered : ITcVnContainer;
fbCondition : FB_ConditionMoreThanFive;
nSelectedContainers : ULINT;
hr : HRESULT;
i : INT;
END_VAR
```
### aContainerStructure
Definition of the container structure. An array of the length 5 with the initialization values [2, 7, 3, 5, 12] means, for example, that a container with 5 sub-containers is to be created in the sample. The first sub-container is to have 2 elements, the second sub-container 7 elements and so on. The length of the array is defined by the constant cNumberOfSubContainers.

### ipHelper
Auxiliary interface pointer that is used to fill the main container ipContainer with sub-containers.

### ipContainerBase
Container in which the sub-containers defined in aNumberOfElement are created.

### fbCondition
Function block that contains the filter condition.

### ipContainerFiltered
Container that, at the end, is to contain only the sub-containers from ipContainer that fulfill the filter condition.

### nSelectedContainers
Integer that indicates how many sub-containers actually fulfill the filter condition.

### hr
Status variable of the type HRESULT [13].

### i
Auxiliary variable

In the program part, a container is created that is filled with the sub-containers defined above. The calling of F_VN_ReserveContainerMemory [710] is optional and only serves to achieve a better performance.

```pascal
hr := F_VN_CreateContainer(ipContainerBase, ContainerType_Vector_Vector_REAL, 0, hr);
hhr := F_VN_ReserveContainerMemory(ipContainerBase, cNumberOfSubContainers, hr);
FOR i:=0 TO (cNumberOfSubContainers-1) DO
  hr := F_VN_CreateContainer(ipHelper, ContainerType_Vector_REAL, aContainerStructure[i], hr);
  hr := F_VN_AppendToContainer_ITcVnContainer(ipHelper, ipContainerBase, hr);
END_FOR
```

The container that has just been created is then filtered. The function block with the self-written condition is specified as the filter condition.

```pascal
hr := F_VN_CopyContainerElementsConditional_ITcVnContainer(
  ipSrcContainer := ipContainerBase,
  ipDestContainer := ipContainerSelection,
  ipConditionFB := fbCondition,
  hr);
```

In order to comprehend the functionality of the function, the elements that fulfill the described condition are counted. The variable nSelectedContainers now changes depending on the sub-containers and the filter condition.

```pascal
hr := F_VN_GetNumberOfElements(ipContainerSelection, nSelectedContainers, hr);
```

In this case the number indicates how many sub-containers have more than five sub-elements.

### Function block with filter condition

The filter condition is transferred as a function block with a Condition method. So that a self-written function block can be transferred to the function, it must implement the interface ITcVnCustomElementCondition_ITcVnContainer [226]:

```pascal
FUNCTION_BLOCK FB_ConditionMoreThanFive IMPLEMENTS ITcVnCustomElementCondition_ITcVnContainer
METHOD Condition : BOOL
  VAR_INPUT
    ipElement : ITcVnContainer;
  END_VAR

  hr := F_VN_GetNumberOfElements(ipElement, nNumberOfElements, hr);
  IF FAILED(hr) OR nNumberOfElements <= 5 THEN
    Condition := FALSE;
  END_IF
END_FUNCTION_BLOCK
```
ELSE
    Condition := TRUE;
END_IF

The required variables are declared as follows:

VAR
    nNumberOfElements : UINTEGER;
    hr                 : HRESULT;
END_VAR

Both the function block itself and the Condition method must contain the compiler attribute {attribute 'c++_compatible'}.

8.1.2 Basic Image Operations

8.1.2.1 Fusing several images

In this sample, five images are joined using the F_VN_FuseImagesArray [734] function. This functionality is particularly useful in conjunction with line scan cameras and seamless recording of continuous material flow.

Explanation

The F_VN_FuseImagesArray [734] function accepts an array of (up to ten) images and joins them in a new image. In the following sample program, initially 5 differently colored images with a height of 10px are created. The number can be changed with the variable nImages and the height with nHeight. Be sure to vary this number in the range 1-10 only.

The size of the result image is determined by specifying the image lines with the parameters nFirstLine and nNumLines. This makes it possible to crop the first and last image.

The function F_VN_FuseImages [733] can also be used if only two images are to be joined.

Program

The following variables are declared:

VAR
    hr : HRESULT;
    aImages : ARRAY[0..9] OF ITcVnImage;
    ipImageFused : ITcVnImage;
    ipImageFusedDisp : ITcVnDisplayableImage;
    nImages : INT := 5;
    nHeight : UDINT := 10;
    aColor : ARRAY[0..9] OF TcVnVector4_LREAL := [[255,0,0], [200,50,0], [150,100,0], [100,150,0], [50,200,0], [0,250,0], [0,200,50], [0,150,100], [0,100,150], [0,50,200]];
    i : INT;
END_VAR

In the program the images in the array aImages are first of all initialized and colored. In order to be able to visually tell the images apart, ten different colors from aColor are used.
FOR i := 0 TO nImages-1 DO
   hr := F_VN_CreateImageAndSetPixels(aImages[i], 100, nHeight, ETcVnElementType.TCVN_ET_USINT, 3, aColor[i], hr);
END_FOR

After that a defined number nImages of images from aImages are joined vertically in the image ipImageFused:
hr := F_VN_FuseImagesArray(aImages, nImages, ipImageFused, 0, nImages * nHeight, hr);

Finally, the image is displayed:
hr := F_VN_TransformIntoDisplayableImage(ipImageFused, ipImageFusedDisp, hr);

Result

nImages = 2:

nImages = 5:

nImages = 10:

Similar samples

Copy image areas [▶ 1367]

8.1.2.2 Copy image areas

In this sample, two images are copied to another image using different methods:

- Using the function F_VN_CopyImageRegionToRegion [▶ 720],
- Selecting a ROI and copying an image, and
- Selecting a ROI and adding two images using the function F_VN_AddImages [▶ 468].

Explanation

The methods shown below can be used to swap rectangular image areas between images. If required, this can also be done in a partially transparent manner with the aid of arithmetic image operations.

The function F_VN_CopyImageRegionToRegion [▶ 720] takes two images and two image regions and copies the contents of one image region to the other. Both image regions have the same size and must be fully present in the image. The function is used in the sample to copy the red image to the mixed image. However, to copy the blue image, a programmed version with the same functionality is used. The advantage of this version is that instead of copying, it is also possible to mix the blue and the mixed image.

In the sample, the position of the two copied images can be changed using the variables aPositionRed and aPositionBlue. The parameter bCopyBlue determines whether the blue image is copied or mixed.
Variables

hr : HRESULT;

ipImageMerge : ITCvNImage;
ipImageRed : ITCvNImage;
ipImageBlue : ITCvNImage;
ipImageMergeDisp : ITCvNDisplayableImage;

aBlack : TcVnVector4_LREAL := [0, 0, 0];
aRed : TcVnVector4_LREAL := [255, 0, 0];
aBlue : TcVnVector4_LREAL := [0, 0, 255];

aPositionRed : TcVnPoint := [20, 20];
aPositionBlue : TcVnPoint := [480, 480];
bCopyBlue : BOOL := FALSE;

Program

// create images & set colors
hr := F_VN_CreateImageAndSetPixels(ipImageMerge, 1000, 1000, TCVN_ET_USINT, 3, aBlack, hr);
hr := F_VN_CreateImageAndSetPixels(ipImageRed, 500, 500, TCVN_ET_USINT, 3, aRed, hr);
hr := F_VN_CreateImageAndSetPixels(ipImageBlue, 500, 500, TCVN_ET_USINT, 3, aBlue, hr);

// Copy red image to selected region in merge-image
hr := F_VN_CopyImageRegionToRegion(
    ipSrcImage := ipImageRed,
    nXSrc := 0, nYSrc := 0,
    nWidth := 500, nHeight := 500,
    ipDestImage := ipImageMerge,
    nXDest := aPositionRed[0],
    nYDest := aPositionRed[1],
    hrPrev := hr
);

// Copy or mix blue image to selected region in merge-image
hr := F_VN_SetRoi(aPositionBlue[0], aPositionBlue[1], 500, 500, ipImageMerge, hr);
IF bCopyBlue THEN
    hr := F_VN_CopyImage(ipImageBlue, ipImageMerge, hr);
ELSE
    hr := F_VN_AddImages(ipImageBlue, ipImageMerge, ipImageMerge, hr);
END_IF
hr := F_VN_ResetRoi(ipImageMerge, hr);
hr := F_VN_TransformIntoDisplayableImage(ipImageMerge, ipImageMergeDisp, hr);

Result

The following are samples of the program results. In the images on the left and in the center, the blue square was added, in the image on the right it was copied. In addition, the target positions of the squares are different.

Similar samples

Fusing several images [】1366]
8.1.3 Code Reading

8.1.3.1 EAN-13 Barcode Reading

This sample illustrates the following:

- One EAN13 code is read per image using the function `F_VN_ReadBarcodeExp`.
- The execution time of the function is monitored via a watchdog.

Explanation:

- The functions `F_VN_ReadBarcode` / `F_VN_ReadBarcodeExp` can search and read barcodes of different types in the image. To speed up the processing time, the barcode type (`eBarcodeType`) can be specified. In this example, these are `TCVN_BT_EAN13`. In addition, the processing time can be monitored via a watchdog. The sample `Blob Detection with watchdog monitoring` provides more comprehensive information on the watchdog.
- Compared to `F_VN_ReadBarcode`, the function `F_VN_ReadBarcodeExp`, which is used in this example, has the following features:
  - in addition to the barcode type, the search direction (`eSearchDirection`) can be specified:
    - `TCVN_BSD_ANY` first searches in horizontal direction and then in vertical direction
    - `TCVN_BSD_HORIZONTAL` only searches in horizontal direction
    - `TCVN_BSD_VERTICAL` only searches in vertical direction
  - the code section in which the readout took place can be returned.

Variables:

```
hr : HRESULT;
ipImageIn : ITcVnImage;
ipImageInDsply : ITcVnDisplayableImage;
ipImageRes : ITcVnImage;
ipImageResDsply : ITcVnDisplayableImage;
// Barcode
ipCodeDecodedList : ITcVnContainer;
ipCodeContourList : ITcVnContainer;
sCodeAsString : STRING(255);
eBarcodeSearchDirection : ETcVnBarcodeSearchDirection := TCVN_BSD_ANY;
eBarcodeType : ETcVnBarcodeType := TCVN_BT_EAN13;
// Watchdog
hrWD : HRESULT;
tStop : DINT := 50000;
tRest : DINT;
// Output
sText : STRING;
// Color
aColorRed : TcVnVector4_LREAL := [255, 0, 0];
```

Code:

```c
// Execute the Barcode Reading Function with EAN13 selected monitored by the Watchdog-Function
hr := F_VN_ReadBarcodeExp(ipSrcImage := ipImageIn,
ipDecodedData := ipCodeDecodedList,
ipContours := ipCodeContourList,
eBarcodeType := eBarcodeType,
nCodeNumber := 1,
eSearchDirection := eBarcodeSearchDirection,
hrPrev := hr);
hrWD := F_VN_StopWatchdog(hrWD, tRest => tRest);
```

// Check if the function was executed successfully
IF hr = S_OK THEN
  // Export Code into String
  hr := F_VN_ExportSubContainer_String(ipCodeDecodedList, 0, sCodeAsStr, 255, hr);
  // Write Code into Result Image
  hr := F_VN_PutTextExp(sCodeAsStr, ipImageRes, 50, 100, ETcVnFontType.TCVN_FT_HERSHEY.PLAIN, 5, aColorRed, 3, TCVN_LT_4_CONNECTED, FALSE, hr);
  // Draw Code Contour into Result Image
  hr := F_VN_DrawContours(ipCodeContourList, 0, ipImageRes, aColorRed, 3, hr);
ELSE
  // Write HRESULT into Result Image
  sText := CONCAT('Returncode ', DINT_TO_STRING(hr));
  hr := F_VN_PutTextExp(sText, ipImageRes, 50, 100, ETcVnFontType.TCVN_FT_HERSHEY.PLAIN, 5, aColorRed, 3, TCVN_LT_4_CONNECTED, FALSE, hr);
END_IF

// Write Code Reading proceeded time into Result Image
sText := CONCAT(CONCAT('Time: ', DINT_TO_STRING(tStop - tRest)), 'us');
hr := F_VN_PutTextExp(sText, ipImageRes, 50, 200, ETcVnFontType.TCVN_FT_HERSHEY.PLAIN, 5, aColorRed, 3, TCVN_LT_4_CONNECTED, FALSE, hr);

Results

If TCVN_BSD_ANY is selected as the search direction, both horizontal and vertical EAN13 codes will be detected. When comparing the watchdog time, however, it becomes apparent that horizontal EAN13 codes (first image) are recognized much faster than vertical EAN13 codes (second image), since the code search initially takes place in horizontal direction.
If the search direction is set to TCVN_BSD_VERTICAL, the vertical EAN13 code is recognized much faster (third image). However, in this case there is no search for horizontal codes.

Similar samples

- Data Matrix Code Reading [► 1372]
- Pharma-Code Reading [► 1373]
- QR-Code Reading [► 1375]
8.1.3.2 Data Matrix Code Reading

This sample illustrates the following

- For each image a Data Matrix Code is read using the function `F_VN_ReadDataMatrixCodeExp`.
- The execution time of the function is monitored via a watchdog.

Explanation

The function `F_VN_ReadDataMatrixCodeExp` searches for Data Matrix Codes in the specified image and reads them. In contrast to the `F_VN_ReadDataMatrixCode` function, the number of codes to be searched for each image can be specified. In this sample, it is one code per image. In addition, the contour of the code that is found is returned. `TCVN_CSS_ONLY_NOT_FLIPPED` is selected as the search strategy, since the image is not mirrored.

If there are areas in the image that look similar to codes, or if there is no code at all in the image, this can have a negative effect on the execution time. To prevent cycle time overruns, in this sample the function `F_VN_ReadDataMatrixCodeExp` is monitored with the aid of a watchdog (see the sample on Blob Detection with watchdog monitoring).

Variables

```plaintext
hr : HRESULT;
ipImageIn : ITcVnImage;
ipImageInDisp : ITcVnDisplayableImage;
ipImageRes : ITcVnImage;
ipImageResDisp : ITcVnDisplayableImage;

// Data Matrix Code
ipCodeDecodedList : ITcVnContainer;
ipCodeContourList : ITcVnContainer;
sCodeAsString : STRING(255);

// Watchdog
hrWD : HRESULT;
tStop : DINT := 20000;
tRest : DINT;

// Output
sText : STRING;

// Color
aColorRed : TcVnVector4_LREAL := [255, 0, 0];
```

Code

```plaintext
// Execute the DMC Code Reading Function monitored by the Watchdog function
// ---------------------------------------------
hrWD := F_VN_StartRelWatchdog(tStop, S_OK);

hr := F_VN_ReadDataMatrixCodeExp(
    ipSrcImage := ipImageIn,
    ipDecodedData := ipCodeDecodedList,
    ipContours := ipCodeContourList,
    nCodeNumber := 1,
    eSearchStrategy := TCVN_CSS_ONLY_NOT_FLIPPED,
    hrPrev := hr
);

hrWD := F_VN_StopWatchdog(hrWD, tRest => tRest);

// Check if the function was executed successfully
IF hr = S_OK THEN
    // Export Code into String
    hr := F_VN_ExportSubContainer_String(ipCodeDecodedList, 0, sCodeAsString, 255, hr);

    // Write Code into Result Image
    hr := F_VN_PutText(sCodeAsString, ipImageRes, 25, 50, ETcVnFontType.TCVN_FT_HERSHEY_PLAIN, 2,
                       aColorRed, hr);
```
// Draw Code Contour into Result Image
hr := F_VN_DrawContours(ipCodeContourList, 0, ipImageRes, aColorRed, 3, hr);
ELSE
    // Write HRESULT into Result Image
    sText := CONCAT('Returncode ', DINT_TO_STRING(hr));
    hr := F_VN_PutLabelExp(sText, ipImageRes, 25, 50, 2, 2, ETcVnFontType.TCVN_FT_HERSHEY_PLAIN, aColorRed, aColorWhite, ETcVnLineType.TCVN_LT_8_CONNECTED, hr);
END_IF

// Write Code Reading proceeded time into Result Image
sText := CONCAT(CONCAT('Time: ', DINT_TO_STRING(tStop - tRest)), 'us');
hr := F_VN_PutText(sText, ipImageRes, 25, 80, ETcVnFontType.TCVN_FT_HERSHEY_PLAIN, 2, aColorRed, hr);

Result

**TwinCAT**

**Time: 17008us**

---

**Similar samples**

- **EAN-13 Barcode Reading [1369]**
- **Pharma-Code Reading [1373]**
- **QR-Code Reading [1375]**
- **Result evaluation during Code Reading [1377]**

**8.1.3.3 Pharma-Code Reading**

This sample illustrates the following

- One pharma code is read per image using the function `F_VN_ReadPharmaCodeExp [764]`
- The execution time of the function is monitored via a watchdog [137].
Samples

Explanation

- The function `F_VN_ReadPharmaCodeExp [764]` searches for pharma codes in the specified image and reads them. In contrast to the function `F_VN_ReadPharmaCode [762]`, the code position is also returned.

- Code-like structures have a negative effect on the execution time. To prevent cycle time overruns, in the sample the function `F_VN_ReadPharmaCodeExp [764]` is monitored via a watchdog [137]. The sample Blob Detection with watchdog monitoring [1390] explains how to use a watchdog [137].

Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>hr</td>
<td>HRESULT</td>
</tr>
<tr>
<td>ipImageIn</td>
<td>ITcVnImage</td>
</tr>
<tr>
<td>ipImageInDisp</td>
<td>ITcVnDisplayableImage</td>
</tr>
<tr>
<td>ipImageRes</td>
<td>ITcVnImage</td>
</tr>
<tr>
<td>ipImageResDisp</td>
<td>ITcVnDisplayableImage</td>
</tr>
<tr>
<td>ipCodeDecodedList</td>
<td>ITcVnContainer</td>
</tr>
<tr>
<td>ipCodeContourList</td>
<td>ITcVnContainer</td>
</tr>
<tr>
<td>sCodeAsString</td>
<td>STRING(255)</td>
</tr>
<tr>
<td>aCodeContour</td>
<td>ARRAY[0..3] OF TcVnPoint2_DINT</td>
</tr>
<tr>
<td>hrWD</td>
<td>HRESULT</td>
</tr>
<tr>
<td>tStop</td>
<td>DINT := 5000</td>
</tr>
<tr>
<td>tRest</td>
<td>DINT</td>
</tr>
<tr>
<td>sText</td>
<td>STRING</td>
</tr>
<tr>
<td>aColorRed</td>
<td>TcVnVector4_LREAL := [255, 0, 0]</td>
</tr>
</tbody>
</table>

Code

// Execute the Pharma-Code Reading Function monitored by the Watchdog-Function

hr := F_VN_StartRelWatchdog(tStop, S_OK);
hr := F_VN_ReadPharmaCodeExp(ipSrcImage := ipImageIn,
ipDecodedData := ipCodeDecodedList,
ipContours := ipCodeContourList,
nCodeNumber := 1,
nMinBarNumber := 4,
hrPrev := hr);
hrWD := F_VN_StopWatchdog(hrWD, tRest => tRest);

// Check if the function was executed successfully
IF hr = S_OK THEN
  // Write Code into String
  sText := CONCAT('Code: ', sCodeAsString);
  hr := F_VN_PutTextExp(sText, ipImageRes, 25, 50, TCVN_FT_HERSHEY_PLAIN, 3,
aColorRed, 2, TCVN_LT_4_CONNECTED, FALSE, hr);

  // Draw Code Contour into Result Image
  hr := F_VN_DrawContours(ipCodeContourList, 0, ipImageRes, aColorRed, 3, hr);
ELSE
  // Write HRESULT into Result Image
  sText := CONCAT('Returncode ', DINT_TO_STRING(hr));
  hr := F_VN_PutTextExp(sText, ipImageRes, 25, 50, TCVN_FT_HERSHEY_PLAIN, 3,
aColorRed, 2, TCVN_LT_4_CONNECTED, FALSE, hr);
END_IF

// Write Code Reading proceeded time into Result Image
sText := CONCAT(CONCAT('Time: ', DINT_TO_STRING(tStop - tRest)), ', 'us');
hr := F_VN_PutTextExp(sText, ipImageRes, 25, 100, TCVN_FT_HERSHEY_PLAIN, 3,
aColorRed, 2, TCVN_LT_4_CONNECTED, FALSE, hr);
Result

Code: 1732
Time: 1284us

Similar samples

- Data Matrix Code Reading [1372]
- EAN-13 Barcode Reading [1369]
- QR-Code Reading [1375]
- Result evaluation during Code Reading [1377]

8.1.3.4 QR-Code Reading

This sample illustrates the following

- One QR code is read per image using the function F_VN_ReadBarcodeExp [768]
- The execution time of the function is monitored via a watchdog [137].

Explanation

The function F_VN_ReadPharmaCodeExp [768] searches for QR codes in the specified image and reads them. In contrast to the function F_VN_ReadQRCode [766], the contour of the code that is found is also returned.

If there are areas in the image that look similar to codes, or if there is no code at all in the image, this can have a negative effect on the execution time. To prevent cycle time overruns, in this sample the function F_VN_ReadQRCodeExp [768] is monitored with the aid of a watchdog [137] (see the sample on Blob Detection with watchdog monitoring [1390]).
Variables

hr : HRESULT;
ipImageIn : ITCvNImage;
ipImageInDisp : ITCvNDisplayableImage;
ipImageRes : ITCvNImage;
ipImageResDisp : ITCvNDisplayableImage;

// QR Code
ipCodeDecodedList : ITCvNContainer;
ipCodeContourList : ITCvNContainer;
sCodeAsString : STRING(255);

// Watchdog
hrWD : HRESULT;
tStop : DINT := 20000;
tRest :

// Output
sText :

// Color
aColorRed : TcVnVector4_LREAL := [255, 0, 0];

Code

// Execute the QR Code Reading Function monitored by the Watchdog-Function
// ---------------------------------------------------------------
hr := F_VN_ReadQRCodeExp(
    ipSrcImage := ipImageIn,
ipDecodedData := ipCodeDecodedList,
ipContours := ipCodeContourList,
nCodeNumber := 1,
eSearchStrategy := TCVN_CSS_ONLY_NOT_INVERTED + TCVN_CSS_ONLY_NOT_FLIPPED,
hrPrev := hr
);
hrWD := F_VN_StopWatchdog(hrWD, tRest => tRest);

// Check if the function was executed successfully
IF hr = S_OK THEN
    // Export Code into String
    hr := F_VN_ExportSubContainer_String(ipCodeDecodedList, 0, sCodeAsString, 255, hr);

    // Write Code into Result Image
    hr := F_VN_PutTextExp(sCodeAsString, ipImageRes, 50, 100, ETcVnFontType.TCVN_FT_HERSHEY_PLAIN, 5, aColorRed,3, TCVN_LT_4_CONNECTED, FALSE, hr);

    // Draw Code Contour into Result Image
    hr := F_VN_DrawContours(ipCodeContourList, 0, ipImageRes, aColorRed, 3, hr);
ELSE
    // Write HRESULT into Result Image
    sText := CONCAT('Returncode ', DINT_TO_STRING(hr));
    hr := F_VN_PutTextExp(sText, ipImageRes, 50, 100, ETcVnFontType.TCVN_FT_HERSHEY_PLAIN, 5, aColorRed,3, TCVN_LT_4_CONNECTED, FALSE, hr);
END_IF

// Write Code Reading proceeded time into Result Image
sText := CONCAT(CONCAT('Time: ', DINT_TO_STRING(tStop - tRest)), 'us'));
hr := F_VN_PutTextExp(sText, ipImageRes, 50, 200, ETcVnFontType.TCVN_FT_HERSHEY_PLAIN, 5, aColorRed,3, TCVN_LT_4_CONNECTED, FALSE, hr);
Samples

Result

TwinCAT Vision
Time: 7207us

Similar samples

• Data Matrix Code Reading [1372]
• EAN-13 Barcode Reading [1369]
• Pharma-Code Reading [1373]
• Result evaluation during Code Reading [1377]

8.1.3.5 Result evaluation during Code Reading

This sample illustrates the following

• One EAN13 code is read per image using the function F_VN_ReadBarcode [756]
• The execution time of the function is monitored via a watchdog [137].
• Handling of different return codes of the function.

Explanation

• The function F_VN_ReadBarcode [755] can search and read barcodes of different types in the image. Details of the use of the function can be found in the sample EAN-13 Barcode Reading [1369].
• The handling of the return code of the function is explained using a corresponding query. A distinction is made between the following cases:
  ◦ Successful execution of the function, code was found and decoded.
  ◦ Successful execution of the function, code is not present in the image or cannot be found / decoded.
  ◦ The function itself has no error and was aborted by the watchdog.
  ◦ Function has other error.

You can find a detailed explanation of the return codes for Code Reading functions in the overview chapter Code Reading [753].
Variables

hr : HRESULT;

// Images
ipImageIn : ITcVnImage;
ipImageInDisp : ITcVnDisplayableImage;
ipImageRes : ITcVnImage;
ipImageResDisp : ITcVnDisplayableImage;

// Barcode
ipCodeDecodedList : ITcVnContainer;
ipCodeContourList : ITcVnContainer;
sCodeAsString : STRING(255);

// Watchdog
hrWD : HRESULT;
tStop : DINT := 50000;
tRest : DINT;

// Color
aColorGreen : TcVnVector4_LREAL := [0, 255, 0];

// Return code
nReturnCode : DWORD;
sReturnCode : STRING;
sResultText : STRING;

Code

// Execute the Barcode Reading Function with EAN13 selected monitored by the Watchdog-Function
hrWD := F_VN_StartRelWatchdog(tStop, S_OK);
hr := F_VN_ReadBarcode(
ipSrcImage := ipImageIn,
ipDecodedData := ipCodeDecodedList,
eBarcodeType := TCVN_BT_EAN13,
hrPrev := hr);
hrWD := F_VN_StopWatchdog(hrWD, tRest => tRest);

// Handle return-code
IF hr = S_OK THEN
// Prepare code result in string
hr := F_VN_ExportSubContainer_String(ipCodeDecodedList, 0, sCodeAsString, 255, hr);
sResultText := CONCAT('Code: ', sCodeAsString);
ELSE
// Check for succeeded return codes or add specific error handling
CASE hr OF
S_FALSE:
sResultText := 'No code found...';
S_WATCHDOG_TIMEOUT:
sResultText := 'Cancelled by watchdog...';
ELSE
// Extract error-code from HRESULT & react accordingly
nReturnCode := DINT_TO_DWORD(hr) AND 16#FFF;
sReturnCode := DWORD_TO_HEXSTR(nReturnCode, 3, FALSE);
sResultText := CONCAT('Returncode ', sReturnCode);
END_CASE
END_IF

// Draw result image
hr := F_VN_PutTextExp(sResultText, ipImageRes, 50, 100, TCVN_FT_HERSHEY_PLAIN, 4,
aColorGreen, 3, TCVN_LT_4_CONNECTED, FALSE, S_OK);
hr := F_VN_TransformIntoDisplayableImage(ipImageRes, ipImageResDisp, hr);

NOTE

Reserve hr

Make sure not to use the variable hr between the call of the Code Reading function and the evaluation as a return value of another function.

Results

If a code is found as expected, is hr = S_OK and the result can be output.
If the expected code cannot be found, the return code is $\text{S\_FALSE}$. If execution is aborted by a watchdog [137], the code is \texttt{16#256}. A corresponding message can be output.

If the function is not executed correctly, the error code can be output. This does not occur in the sample and is only caused by incorrect use of Code Reading functions.

**Similar samples**

- Data Matrix Code Reading [1372]
- EAN-13 Barcode Reading [1369]
- Pharma-Code Reading [1373]
- QR-Code Reading [1375]
8.1.4 Contour analysis

8.1.4.1 Match Contours 1vsN (manual shapes)

In this sample, contours are compared with reference contours using the function F_VN_MatchContours1vsN [873]. Where

- the reference contours are idealized, manually created polygons
- and the other contours are extracted from an image.

Explanation

The function F_VN_MatchContours [872] compares two contours used in the sample based on the Hu moments. The function F_VN_MatchContours1vsN [873] represents an extension that can be used to compare a contour simultaneously with several other contours. In this sample, this is used to classify workpieces according to predefined reference shapes.

The reference shapes are idealized, manually created polygons of an isosceles triangle, a rectangle/square and a trapezoid. Since only the corner points were specified for the polygons, the reference shapes differ greatly from the test contours in the number of descriptive contour points. Furthermore, they differ in position, orientation, size and the degree of interference within the contour.

The function F_VN_DetectBlobs [1029] is used to search for contours with a certain minimum size in order to extract the shape of the components from the image. The contours that are found are then used for the described matching.

Input parameters

In addition to the contours to be compared, only a parameter hast to be passed to the function that describes the calculation method used to determine the dissimilarity from the Hu moments (eComparisonMethod of type FtcVnContoursMatchComparisonMethod [180]). The method TCVN_CMCM_CONTOURS_MATCH_I1 calculates the sum of the differences in the reciprocal values of the individual characteristics, while the method TCVN_CMCM_CONTOURS_MATCH_I2 calculates the sum of the differences in the pure characteristics. Unlike the first two methods, the third variant TCVN_CMCM_CONTOURS_MATCH_I3 only calculates the maximum difference between the individual characteristics. Which of the three methods is best suited for comparing two contours depends on the use case.

Variables

The CustomTcVnArray4_Point2_DINT data type is not included in TwinCAT Vision, but was created explicitly for this sample. The implementation can be found in the download of this sample.

```plaintext
// Indexes representing the different shapes
TRIANGLE : USINT := 0;
RECTANGLE : USINT := 1;
TRAPEZOID : USINT := 2;

// Images
ipImageIn : ITCvnImage;
ipImageInDisp : ITCvnDisplayableImage;
ipImageRes : ITCvnImage;
ipImageResDisp : ITCvnDisplayableImage;

// Contours
ipContourList : ITCvnContainer;
ipContour : ITCvnContainer;
ipIterator : ITCvnForwardIterator;
aShapeArrays : ARRAY [0..2] OF CustomTcVnArray4_Point2_DINT :=
    [[0,0],[40,0],[20, LREAL_TO_DINT(SQRT(1200))], [0,0]], // Triangle
    [[0,0],[40,0],[40,40],[0,40]], // Rectangle
    [[0,0],[40,0],[60,22],[20,22]]]; // Trapezoid;
aShapes : ARRAY[0..2] OF ITCvnContainer;

ipShapes : ITCvnContainer;
```
// Matching
ipMatchIndexes : ITcVnContainer;
ipDissimilarities : ITcVnContainer;
aMatchIndexes : ARRAY [0..2] OF ULINT;
aDissimilarities : ARRAY [0..2] OF LREAL;

// Parameters
stBlobParams : TcVnParamsBlobDetection;
fThreshold : REAL := 170;
fMinArea : REAL := 10000;
fMaxDissimilarity : LREAL := 0.02;
eComparisonMethod : ETcVnContoursMatchComparisonMethod := TCVN_CMCM_CONTOURS_MATCH_I3;

// drawing
aColors : ARRAY[0..2] OF TcVnVector4_LREAL := 
[ [0, 175, 0, 0], [0, 0, 255, 0], [255, 0, 0, 0] ]; // green, blue, red
aColorWhite : TcVnVector4_LREAL := [255, 255, 255];
aColorBlack : TcVnVector4_LREAL := [0, 0, 0];
aTexts : ARRAY[0..2] OF STRING := ['Triangle', 'Rectangle', 'Trapezoid'];
sText : STRING(255); nTopLeftX : UDINT; nTopLeftY : UDINT;

// Miscellaneous
aPixelValue : TcVnVector4_LREAL;
I : USINT;
stBoundingRectangle : TcVnRectangle_UDINT;
aOffsets : ARRAY[0..2] OF TcVnPoint := 
[ [20, 50], [20, 100], [20, 160] ];
hr : HRESULT;

Code

// Fill manually defined shapes into one container
hr := F_VN_CreateContainer(ipShapes, ContainerType_Vector_Vector_TcVnPoint2_DINT, 0, hr);
FOR i:=TRIANGLE TO TRAPEZOID DO
    hr := F_VN_CreateContainerFromArray(ADR(aShapeArrays[i]), aShapes[i], ContainerType_Vector_TcVnPoint2_DINT, 4, hr);
    hr := F_VN_InsertIntoContainer_ITcVnContainer(aShapes[i], ipShapes, i, hr);
END_FOR

// Prepare result image
hr := F_VN_ConvertColorSpace(ipImageIn, ipImageRes, TCVN_CST_GRAY_TO_RGB, hr);

// Check if background is light or dark
hr := F_VN_GetPixel(ipImageIn, aPixelValue, 50, 50, hr);
IF SUCCEEDED(hr) AND_THEN aPixelValue[0] < 128 THEN
    stBlobParams.eThresholdType := TCVN_TT_BINARY;
ELSE
    stBlobParams.eThresholdType := TCVN_TT_BINARY_INV;
END_IF

// Find contours in image
stBlobParams.bFilterByArea := TRUE;
stBlobParams.fMinArea := fMinArea;
stBlobParams.fMaxDissimilarity := fMaxDissimilarity;
hr := F_VN_DetectBlobs(ipImageIn, ipContourList, stBlobParams, hr);

// Iterate through all found contours
hr := F_VN_GetForwardIterator(ipContourList, ipIterator, hr);
WHILE hr = S_OK AND_THEN ipIterator.CheckIfEnd() <> S_OK DO
    hr := F_VN_GetContainer(ipIterator, ipContour, hr);
    hr := F_VN_IncrementIterator(ipIterator, hr);
    hr := F_VN_MatchContours1vsN(
        ipRefContour:= ipContour,
        ipContours:= ipShapes,
        ipMatchIndexes:= ipMatchIndexes,
        ipDissimilarities:= ipDissimilarities,
        fDissimilarityThreshold:= 100,
        eComparisonMethod:= eComparisonMethod,
        hrPrev:= hr
    );
    hr := F_VN_ExportContainer(ipMatchIndexes, ADR(aMatchIndexes), SIZEOF(aMatchIndexes), hr);
END_WHILE
hr := F_VN_ExportContainer(ipDissimilarities, ADR(aDissimilarities), SIZEOF(aDissimilarities), hr);

// Draw matching results
IF aDissimilarities[0] < fMaxDissimilarity THEN

    // Calculate position of object
    hr := F_VN_UprightBoundingRectangle(ipContour, stBoundingRectangle, hr);
    nTopLeftX := LREAL_TO_UDINT(stBoundingRectangle.nX + 30);
    nTopLeftY := LREAL_TO_UDINT(stBoundingRectangle.nY + (stBoundingRectangle.nHeight / 2));

    // Draw matching result
    hr := F_VN_DrawContours(ipContour, -1, ipImageRes, aColors[aMatchIndexes[0]], 5, hr);
    hr := F_VN_PutTextExp(aTexts[aMatchIndexes[0]], ipImageRes, nTopLeftX, nTopLeftY,
    TCVN_FT_HERSHEY_SIMPLEX, 0.8, aColors[aMatchIndexes[0]], 2, TCVN_LT_8_CONNECTED, FALSE, hr);
    hr := F_VN_DrawRectangle(nTopLeftX, nTopLeftY + 5, nTopLeftX + 200, nTopLeftY + 85,
    ipImageRes, aColorBlack, -1, hr);
    // Draw all dissimilarity values
    FOR i:=0 TO 2 DO
        sText := CONCAT(LEFT(aTexts[aMatchIndexes[i]], 4), CONCAT(' ',
        REAL_TO_STRING(LREAL_TO_REAL(aDissimilarities[i]))));
        hr := F_VN_PutTextExp(sText, ipImageRes, nTopLeftX + 5, nTopLeftY + 30 + 20*i,
        TCVN_FT_HERSHEY_SIMPLEX, 0.6, aColorWhite, 1, TCVN_LT_8_CONNECTED, FALSE, hr);
    END_FOR

END_IF
END_WHILE

// Draw reference shapes
FOR i:=TRIANGLE TO TRAPEZOID DO
    hr := F_VN_DrawContoursExp(aShapes[i], -1, ipImageRes, aColors[i], 2, TCVN_LT_8_CONNECTED, 0, 0,
    aOffsets[i], hr);
END_FOR
hr := F_VN_TransformIntoDisplayableImage(ipImageRes, ipImageResDisp, hr);
hr := F_VN_TransformIntoDisplayableImage(ipImageIn, ipImageInDisp, hr);

Results

The result image shows the reference shapes (left border) and the found contours whose dissimilarity
relating to all reference shapes does not exceed a maximum value. Furthermore, a visualization indicates
which reference shape the contour resembles most closely. A corresponding description text (including
dissimilarities for each reference shape) is superimposed over the respective contour. For the parameters
used in this sample, the result for the MatchContours1.bmp image looks as follows:
In each case the lowest value indicates the correct reference contour. If the parameter eComparisonMethod is changed from TCVN_CMCM_CONTOURS_MATCH_I3 to TCVN_CMCM_CONTOURS_MATCH_I2, the results change considerably. This also means that the parameter for the maximum accepted dissimilarity must be adjusted:
The new values allow a much less clear assignment to corresponding reference shapes, and the round component is incorrectly assigned to the rectangle reference shape. This calculation method is therefore obviously less suitable for this sample.

**Similar samples**
- Match Contours (extracted shapes) [1384]

### 8.1.4.2 Match Contours (extracted shapes)

In this sample, two contours are compared using the function `F_VN_MatchContours` [872].

**Explanation**

The function `F_VN_MatchContours` [872] compares two contours used in the sample based on the Hu moments. In this sample, the function is used to recognize a given component in a new situation. For this purpose, a component is selected in a reference image, its contour is saved and then compared with the contours from another image. In this way, the components that fall below a certain dissimilarity compared to the reference component are marked.

**Input parameters**

In addition to the contours to be compared, only a parameter has to be passed to the function that describes the calculation method used to determine the dissimilarity from the Hu moments (`eComparisonMethod` of type `ETcVnContoursMatchComparisonMethod` [180]). The method `TCVN_CMCM_CONTOURS_MATCH_I1` calculates the sum of the differences in the reciprocal values of the individual characteristics, while the method `TCVN_CMCM_CONTOURS_MATCH_I2` calculates the sum of the differences in the pure characteristics. Unlike the first two methods, the third variant...
TCVN_CMCM_CONTOURS_MATCH_I3 only calculates the maximum difference between the individual characteristics. Which of the three methods is best suited for comparing two contours depends on the use case.

Variables

```
hr : HRESULT;
ipImageIn : ITCvImage;
ipImageInDisp : ITCvDisplayableImage;
ipImageRes : ITCvImage;
ipImageResDisp : ITCvDisplayableImage;
iterator : ITCvForwardIterator;
// result
ipContourList : ITCvContainer;
ipContourReference : ITCvContainer;
ipContourCheck : ITCvContainer;
fBestDissimilarity : LREAL;
fDissimilarity : LREAL;
// parameters
fThreshold : REAL := 170;
fMinArea : REAL := 10000;
fMaxDissimilarity : LREAL := 0.01;
eComparisonMethod : ETCvContoursMatchComparisonMethod := TCVN_CMCM_CONTOURS_MATCH_I3;
// drawing
aColorGreen : TcVnVector4_LREAL := [0, 175, 0];
aColorBlue : TcVnVector4_LREAL := [0, 0, 255];
aColorRed : TcVnVector4_LREAL := [255, 0, 0];
aColorRes : TcVnVector4_LREAL;
sText : STRING(255);
sTextReference : STRING(255) := 'Reference contour';
sTextCheck : STRING(255) := 'Check dissimilarity';
// other
bDarkBackground : BOOL;
nContours : ULINT;
nCounter : UINT := 0;
aPixelValue : TcVnVector4_LREAL;
stBoundingRectangle : TcVnRectangle_UDINT;
stParams : TcVnParamsBlobDetection;
```

Code

```
// Prepare result image
hr := F_VN_ConvertColorSpace(ipImageIn, ipImageRes, TCVN_CST_GRAY_TO_RGB, hr);

// Check if background is dark in order to identify reference image
hr := F_VN_GetPixel(ipImageIn, aPixelValue, 50, 50, hr);
bDarkBackground := SUCCEEDED(hr) AND_THEN aPixelValue[0] < 128;
IF bDarkBackground THEN
    stParams.eThresholdType := TCVN_TT_BINARY;
ELSE
    stParams.eThresholdType := TCVN_TT_BINARY_INV;
END_IF

// Find contours in image
hr := F_VN_DetectBlobs(ipImageIn, ipContourList, stParams, hr);

// Distinguish reference and test image
IF bDarkBackground THEN
    // Select one of the found contours as reference for matching
    hr := F_VN_GetNumberOfElements(ipContourList, nContours, hr);
    hr := F_VN_GetAt_ITcVnContainer(ipContourList, ipContourReference, nCounter MOD nContours, hr);

    // Draw selected contour and text
    hr := F_VN_PutTextExp(sTextReference, ipImageRes, 50, 50, TCVN_FT_HERSHEY_SIMPLEX, 1.3,
        aColorBlue, 2, TCVN_LT_8_CONNECTED, FALSE, hr);
    hr := F_VN_DrawContours(ipContourReference, -1, ipImageRes, aColorBlue, 5, hr);
    nCounter := nCounter + 1;
ELSE
    ...
fBestDissimilarity := 10E300;

// Iterate through all found contours
hr := F_VN_GetForwardIterator(ipContourList, ipIterator, hr);
WHILE hr = S_OK AND_THEN ipIterator.CheckIfEnd() <> S_OK DO
    hr := F_VN_GetContainer(ipIterator, ipContourCheck, hr);
    hr := F_VN_IncrementIterator(ipIterator, hr);

    // Match the current contour with the selected reference contour
    hr := F_VN_MatchContours(ipContourReference, ipContourCheck, eComparisonMethod, fDissimilarity, hr);

    // Save best result
    IF fBestDissimilarity > fDissimilarity THEN
        fBestDissimilarity := fDissimilarity;
    END_IF

    // Choose action depending on the dissimilarity of both contours
    IF SUCCEEDED(hr) AND_THEN fDissimilarity < fMaxDissimilarity THEN
        aColorRes := aColorGreen;
    ELSE
        aColorRes := aColorRed;
    END_IF

    // Draw matching results
    sText := REAL_TO_STRING(LREAL_TO_REAL(fDissimilarity));
    hr := F_VN_DrawContours(ipContourCheck, -1, ipImageRes, aColorRes, 5, hr);
    hr := F_VN_UprightBoundingRectangle(ipContourCheck, stBoundingRectangle, hr);
    hr := F_VN_PutTextExp(sText, ipImageRes, LREAL_TO_UDINT(stBoundingRectangle.nX + 30),
        LREAL_TO_UDINT(stBoundingRectangle.nY + (stBoundingRectangle.nHeight / 2)), TCVN_FT_HERSHEY_SIMPLEX, 0.8, aColorRes, 2, TCVN_LT_8_CONNECTED, FALSE, hr);

    END_WHILE

END_IF

hr := F_VN_TransformIntoDisplayableImage(ipImageRes, ipImageResDisp, hr);
hr := F_VN_TransformIntoDisplayableImage(ipImageIn, ipImageInDisp, hr);

Results

The currently selected reference contour is marked in blue on the reference image:
You can see that the contour is slightly curved by the lens distortion and has some minor flaws. In the test pattern, contours whose dissimilarity relative to the reference contour falls below the set threshold value are displayed in green, the other contours are shown in red. In addition, the calculated dissimilarity is superimposed over all contours. The following illustration shows the result for the MatchContours1.bmp input imaged when the rectangle is selected as a reference component:
Check dissimilarity
Best match: 0.001019094

0.2005726
0.0015405
0.00149999
0.00149999

It can be seen that the rectangular component has by far the smallest dissimilarity compared to the reference contour. If the parameter eComparisonMethod is changed from TCVN_CMCM_CONTOURS_MATCH_I3 to TCVN_CMCM_CONTOURS_MATCH_I2, the calculated dissimilarities change accordingly:
In this case, the result is even clearer, but this does not apply to all shapes. If the triangle is selected as the reference component, the result with this setting is significantly less clear than with the original setting:
This result demonstrates that different calculation methods can provide the best results, depending on the application.

**Similar sample**
- Match Contours 1vsN (manual shapes) [1380]

**8.1.5 Image analysis**

**8.1.5.1 Object Detection**

**8.1.5.1.1 Blob Detection with watchdog monitoring**

This sample illustrates the following
- The function `F_VN_DetectBlobs [1029]` is used to find round structures in images.
- The execution time of the function is monitored via a watchdog.

**Explanation**

The `F_VN_DetectBlobs [1029]` function facilitates the search for similar structures in the image. It combines setting threshold values with searching and filtering for contours in a single call. In this sample, the function is parameterized to return all round structures in the example images.
Due to external influences, the actual image content can deviate greatly from the expected image content. With image processing algorithms whose execution time depends on the image content, this can lead to cycle time overruns and thus to undesired side effects. To prevent this there is a watchdog. The watchdog can abort functions marked with the sentence: "Can return partial results when canceled by Watchdog" in the API after a specified period of time. The results obtained up to this point can be used further.

F_VN_DetectBlobs is one the functions that can be monitored in this way. By changing the permissible execution time \( t_{\text{Stop}} \) in the watchdog start function \( \text{F_VN_StartRelWatchdog} \) \( \uparrow \text{880} \), a function termination followed by application of partial results can be observed and compared to full execution of F_VN_DetectBlobs. The watchdog can be stopped via \( \text{F_VN_StopWatchdog} \) \( \uparrow \text{882} \). A positive return value in \( t_{\text{Rest}} \) specifies the remaining time, while a negative return value in \( t_{\text{Rest}} \) specifies the additional time required for termination.

**Task configuration**

Watchdog monitoring \( \uparrow \text{137} \) must be enabled in the executing PLC task. Otherwise the watchdog function calls in the PLC are ignored and the start watchdog function returns the return value 16#71A: "No Interface".

**Variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>hr</td>
<td>HRESULT</td>
</tr>
<tr>
<td>hrFunc</td>
<td>HRESULT</td>
</tr>
<tr>
<td>ipImageIn</td>
<td>ITCvNImage</td>
</tr>
<tr>
<td>ipImageInDisp</td>
<td>ITCvNDisplayableImage</td>
</tr>
<tr>
<td>ipImageRes</td>
<td>ITCvNImage</td>
</tr>
<tr>
<td>ipImageResDisp</td>
<td>ITCvNDisplayableImage</td>
</tr>
<tr>
<td>stBlobParams</td>
<td>TcVnParamsBlobDetection</td>
</tr>
<tr>
<td>ipContourList</td>
<td>ITCvNContainer</td>
</tr>
<tr>
<td>hrWD</td>
<td>HRESULT</td>
</tr>
<tr>
<td>tStop</td>
<td>DINT := 5000</td>
</tr>
<tr>
<td>tRest</td>
<td>DINT</td>
</tr>
<tr>
<td>nFraction</td>
<td>UDINT</td>
</tr>
<tr>
<td>sText</td>
<td>STRING(255)</td>
</tr>
</tbody>
</table>
// Color
aColorGreen : TcVnVector4_LREAL := [0, 255, 0, 0];

// Image Infos
stPixelFormat : TcVnPixelFormat;

Code

// Set Parameters
// --------------------------------------------------------------
// - Set fMaxArea to 100000 detect all circles or set it to 10000
to detect only the small circles

stBlobParams.bFilterByArea := TRUE;
stBlobParams.fMinArea := 100;
stBlobParams.fMaxArea := 100000;

stBlobParams.bFilterByCircularity := TRUE;
stBlobParams.fMinCircularity := 0.80;

stBlobParams.fMinThreshold := 70;

// Execute DetectBlobs-Function monitored by Watchdog-Function
// --------------------------------------------------------------
// - Set the Watchdog stop time tStop from 5000us to 1000us to see
// a Watchdog interrupt and that the interim results can be used

hrWD := F_VN_StartRelWatchdog(tStop, S_OK);
hrFunc := F_VN_DetectBlobs(ipSrcImage := ipImageIn,
                          ipBlobContours := ipContourList,
                          stParams := stBlobParams,
                          hrPrev := hr);

hrWD := F_VN_StopWatchdog(hrWD, tRest => tRest, nFractionProcessed => nFraction);

// Draw Result Image
// --------------------------------------------------------------
hr := F_VN_DrawContours(ipContourList, -1, ipImageRes, aColorGreen, 3, hr);

Results

The input image
All circles were found in the image, with the parameterization of the sample code.
Some of the circles were found in the image, by triggering the watchdog after 1 ms. Since the abort process takes some time, the actual time required is greater than 1 ms.

hrWD := F_VN_StartRelWatchdog(tStop:=1000, WATCHDOG_ACCUMULATION_TYPE_MEAN, S_OK);
Only the small circles were found in the image by reducing the maximum enclosing area in pixels.

```c
stBlobParams.fMinArea := 100;
stBlobParams.fMaxArea := 10000;
```
Similar sample

- Find contour instead of Blob Detection [1396]

8.1.5.1.2 Find contour instead of Blob Detection

This sample shows an alternative solution to the Blob Detection sample [1390] using F_VN_FindContoursExp [1040].

Explanation

The function F_VN_DetectBlobs [1029] consolidates many individual image processing steps in one function call. However, the internal processing steps can also be carried out by individual function calls. This can make it easier to set parameters because one can see the intermediate results. Therefore, in this sample the same images are used as in the Blob Detection sample [1390], and the same round structures are searched for.

Variables

hr : HRESULT;
ipImageIn : ITCvImage;
ipImageInDisp : ITCvDisplayableImage;
ipImageWork : ITCvImage;
ipImageThresholdDisp : ITCvDisplayableImage;
ipImageRes : ITCvImage;
ipImageResDisp : ITCvDisplayableImage;

// Sample Specific Variables
ipContourList : ITCvContainer;
ipContourResultList : ITCvContainer;
ipContour : ITCvNContainer;
ipIterator : ITCvNForwardIterator;
aOffset : TcVnPoint;
fThreshold : LREAL := 70;
fArea : LREAL;
fAreaMin : LREAL := 100;
fAreaMax : LREAL := 100000;
fCircularity : LREAL;
fCircularityMin : LREAL := 0.8;
// Image Infos
stPixelFormat : TcVnPixelFormat;
// COLORS
aColorGreen : TcVnVector4_LREAL := [0, 255, 0];

Code

Preprocessing for image segmentation

// Image Segmentation
hr := F_VN_Threshold(ipImageIn, ipImageWork, fThreshold, 255, TCVN_TT_Binary, hr);

Filtering of contours according to the surrounding area in pixels and circularity

// Filter Contours
hr := F_VN_GetForwardIterator(ipContourList, ipIterator, hr);
hr := F_VN_CreateContainer(ipContourResultList, ContainerType_Vector_Vector_TcVnPoint2_DINT, 0, hr);
WHILE SUCCEEDED(hr) AND_THEN ipIterator.CheckIfEnd() <> S_OK DO

hr := F_VN_GetContainer(ipIterator, ipContour, hr);
hr := F_VN_IncrementIterator(ipIterator, hr);

// Filter by Area
hr := F_VN_ContourArea(ipContour, fArea, hr);
IF fArea > fAreaMin AND fArea < fAreaMax THEN

// Filter by Circularity
hr := F_VN_ContourCircularity(ipContour, fCircularity, hr);
IF fCircularity > fCircularityMin THEN

// Add contour to the result contour container
hr := F_VN_AppendToContainer_ITcVnContainer(ipContour, ipContourResultList, hr);
END_IF
END_IF
END_WHILE

Result output

// Draw contours into the result image and display it
hr := F_VN_DrawContours(ipContourResultList, -1, ipImageRes, aColorGreen, 3, hr);
hr := F_VN_TransformIntoDisplayableImage(ipImageRes, ipImageResDisp, hr);

Similar sample

• Blob Detection with watchdog monitoring [1390]

8.1.5.1.3 Find Contour - Approximation Method

This sample deals with the possible approximation modes (eApproximationMethod of type ETcVnContourApproximationMethod[175]) for contour search.
Explanation

The functions \texttt{F\_VN\_FindContours} \cite{1037}, \texttt{F\_VN\_FindContoursExp} \cite{1040} and \texttt{F\_VN\_FindContourByHierarchy} \cite{1033} are used to detect contours in the image. The contours are described by points. Depending on the selected approximation mode (\texttt{eApproximationMethod} of type \texttt{ETcVnContourApproximationMethod} \cite{175}) all points of each contour or a simplified point set are returned:

- **TCVN\_CAM\_NONE**
  All points of the contour are returned

- **TCVN\_CAM\_SIMPLE**
  Loss-free compression is applied

- **TCVN\_CAM\_TC89\_L1** oder **TCVN\_CAM\_TC89\_KCOS**
  Both functions apply one of the variants of the Teh-Chin chain approximation algorithm

Variables

\begin{verbatim}
hr : HRESULT;
ipImageIn : ITcVnImage;
ipImageInDisp : ITcVnDisplayableImage;
ipImageRes : ITcVnImage;
ipImageResDisp : ITcVnDisplayableImage;

// Sample Specific Variables
ipContourList : ITcVnContainer;
ipContour : ITcVnContainer;
ipIterator : ITcVnForwardIterator;
eRetrievalMode : ETcVnContourRetrievalMode := TCVN\_CRM\_LIST;
eApproximationMethod : ETcVnContourApproximationMethod := TCVN\_CAM\_SIMPLE;
aOffset : TcVnPoint;

// Colors
aColorRed : TcVnVector4\_LREAL := [255, 0, 0];
\end{verbatim}

Code

\begin{verbatim}
// Create Result Image
hr := F\_VN\_ConvertColorSpace(ipImageIn, ipImageRes, TCVN\_CST\_Gray\_TO\_RGB, hr);

// Find Contours and their Hierarchy
// -----------------------------------
hr := F\_VN\_FindContoursExp(ipSrcImage := ipImageIn,
ipContours := ipContourList,
eRetrievalMode := eRetrievalMode,
eApproximationMethod := eApproximationMethod,
aOffset := aOffset,
hrPrev := hr);

// Draw the points of the contour
hr := F\_VN\_GetForwardIterator(ipContourList, ipIterator, hr);

WHILE SUCCEEDED(hr) AND\_THEN ipIterator.CheckIfEnd() <> S\_OK DO
hr := F\_VN\_GetContainer(ipIterator, ipContour, hr);
hr := F\_VN\_IncrementIterator(ipIterator, hr);
hr := F\_VN\_DrawPoints(ipContour, ipImageRes, ETcVnDrawShape.TCVN\_DS\_CIRCLE, aColorRed, hr);
END\_WHILE
\end{verbatim}

Results

The input image
The result image with all points drawn according to \texttt{ETcVnContourApproximationMethod}\[\texttt{175}, \texttt{TCVN\_CAM\_NONE}\]

The result image with all points drawn according to \texttt{ETcVnContourApproximationMethod}\[\texttt{175}, \texttt{TCVN\_CAM\_TC89\_L1}\]

Similar samples
- Find Contour - Hierarchy & Retrieval Mode \[\text{1401}\]
- Find contour instead of Blob Detection \[\text{1396}\]
Find Contour - Hierarchy & Retrieval Mode

This sample shows the retrieval mode (ETcVnContourRetrievalMode [178]) for the contour search (F_VN_FindContoursExp [1040] and F_VN_FindContourHierarchyExp [1033]) and how it affects the contour result and the returned hierarchy.

Explanation

The contour retrieval mode (eRetrievalMode of type ETcVnContourRetrievalMode [178]) can be used to specify how the contour hierarchy is taken into account during the contour search. This sample visualizes the results according to the selected mode.

Detailed explanations of the hierarchy structure can be found in F_VN_FindContourHierarchyExp [1033].

Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>hr</td>
<td>HRESULT</td>
</tr>
<tr>
<td>ipImageIn</td>
<td>ITcVnImage</td>
</tr>
<tr>
<td>ipImageInDisp</td>
<td>ITcVnDisplayableImage</td>
</tr>
<tr>
<td>ipImageRes</td>
<td>ITcVnImage</td>
</tr>
<tr>
<td>ipImageResDisp</td>
<td>ITcVnDisplayableImage</td>
</tr>
<tr>
<td>ipImageHierarchy</td>
<td>ITcVnImage</td>
</tr>
<tr>
<td>ipImageHierarchyDisp</td>
<td>ITcVnDisplayableImage</td>
</tr>
</tbody>
</table>

// Sample Specific Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipContourList</td>
<td>ITcVnContainer</td>
</tr>
<tr>
<td>ipContour</td>
<td>ITcVnContainer</td>
</tr>
<tr>
<td>ipHierarchyList</td>
<td>ITcVnContainer</td>
</tr>
<tr>
<td>aHierarchy</td>
<td>TcVnVector4_DINT</td>
</tr>
<tr>
<td>aHierarchyTree</td>
<td>TcVnVector4_DINT</td>
</tr>
<tr>
<td>eRetrievalMode</td>
<td>ETcVnContourRetrievalMode := TCVN_CRM_LIST;</td>
</tr>
<tr>
<td>eApproximationMethod</td>
<td>ETcVnContourApproximationMethod := TCVN_CAM_SIMPLE;</td>
</tr>
<tr>
<td>aOffset</td>
<td>TcVnPoint</td>
</tr>
<tr>
<td>nNumOfElem</td>
<td>ULINT</td>
</tr>
<tr>
<td>nParents</td>
<td>UINT</td>
</tr>
<tr>
<td>i</td>
<td>ULINT</td>
</tr>
<tr>
<td>j</td>
<td>UINT</td>
</tr>
<tr>
<td>stRect</td>
<td>TcVnRectangle_UDINT</td>
</tr>
<tr>
<td>sText</td>
<td>STRING(80)</td>
</tr>
</tbody>
</table>

// Colors

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>aColorWhite</td>
<td>TcVnVector4_LREAL := [255, 255, 255];</td>
</tr>
<tr>
<td>aColorBlack</td>
<td>TcVnVector4_LREAL := [0, 0, 0];</td>
</tr>
<tr>
<td>aColorList</td>
<td>ARRAY[0..4] OF TcVnVector4_LREAL := [255,0,0], [0,255,0], [0,0,255], [200,200,0], [200,0,200];</td>
</tr>
</tbody>
</table>

Code

// Create Result Image
hr := F_VN_ConvertColorSpace(ipImageIn, ipImageRes, TCVN_CST_Gray_TO_RGB, hr);
hn := F_VN_CreateImage(ipImageHierarchy, 130, 200, EtcVnElementType.TCVN_ET_USINT, 3, hr);
hn := F_VN_SetPixels(ipImageHierarchy, aColorWhite, hr);

// Find Contours and their Hierarchy


hr := F_VN_GetNumberOfElements(ipContourList, nNumOfElem, hr);

IF nNumOfElem > 0 THEN
FOR i := 0 TO (nNumOfElem-1) BY 1 DO
    hr := F_VN_GetAt_ITcVnContainer(ipContourList, ipContour, i, hr);
hr := _F_VN_GetAt_TcVnVector4_DINT(ipHierarchyList, aHierarchy, i, hr);

    // Count Parents and set aColor depending on parent number
    nParents := 0;
    aHierarchyTree := aHierarchy;
    WHILE SUCCEEDED(hr) AND_THEN aHierarchyTree[3] >= 0 DO
        nParents := nParents + 1;
        hr := _F_VN_GetAt_TcVnVector4_DINT(ipHierarchyList, aHierarchyTree, aHierarchyTree[3], hr);
    END_WHILE

    // Draw contour and number into the result image
    sText := TO_STRING(i);
    hr := _F_VN_UprightBoundingRectangle(ipContour, stRect, hr);
    hr := _F_VN_PutText(sText, ipImageRes, stRect.nX +10, stRect.nY +15, TCVN_FT_HERSHEY.PLAIN, 1,
                       aColorList[(nParents) MOD 4], hr);
    hr := _F_VN_DrawContours(ipContour, -1, ipImageRes, aColorList[(nParents) MOD 4], 2, hr);

    // Write Hierarchy
    hr := _F_VN_PutText(sText, ipImageHierarchy, 10, TO_UDINT(10 + i*10), TCVN_FT_HERSHEY.PLAIN,
                         0.5, aColorList[(nParents) MOD 4], hr);
    FOR j := 0 TO 3 BY 1 DO
        sText := TO_STRING(aHierarchy[j]);
        hr := _F_VN_PutText(sText, ipImageHierarchy, TO_UDINT(10 + (j+1)*22), TO_UDINT(10 + i*10),
                             TCVN_FT_HERSHEY.PLAIN, 0.5, aColorBlack, hr);
    END_FOR
    END_IF

Results

The input image

The contours found with TCVN_CRM_TREE. The colors render contours at the same level

The hierarchy for the above result image. Each row refers to a contour. The columns have the following meaning:

- Contour index ("colored column")
- Index of the next contour on the same hierarchical level
• Index of the previous contour on the same hierarchical level
• Index of the first child contour
• Index of parent contour

The number -1 indicates that no corresponding contour exists.

```
4  3  -1  1  -1
1  2  -1  -1  4
2  -1  1  -1  4
3  6  4  4  -1
4  -1  -1  5  3
5  -1  -1  -1  4
6  7  3  -1  -1
7  -1  6  8  -1
8  11  -1  9  7
9  -1  -1  14  8
14  -1  -1  -1  9
11  -1  8  12  7
12  13  -1  -1  11
13  14  12  -1  11
14  -1  13  -1  11
```

Similar samples

• Find Contour - Approximation Method [1398]
• Find contour instead of Blob Detection [1396]

8.1.5.1.5 Template Matching with evaluation

In this sample you will perform Template Matching in order to find a component on a printed circuit board. For this purpose, use:

• F_VN_MatchTemplateAndEvaluate [1051]
• F_VN_DrawRectangle [913]
Application

Remember to add both the input image and the template image to the respective File Source. Via the threshold value fMatchThreshold you can set a limit as to how good a match must be in order to be accepted.

Program

Important in this sample is that you require a template image in addition to the normal input image in order to search for occurrences of the template in the input image.

If both images are available, perform the template matching automatically with the function F_VN_MatchTemplateAndEvaluate [1051].

hr := F_VN_MatchTemplateAndEvaluate(ipImageIn, ipImageTemplate, ipMatches, fMatchThreshold, hr);

The matches found whose degree of correlation is higher than the specified threshold value fMatchThreshold are now located as points in the container ipMatches. Each point specifies the top left corner of each occurrence of ipImageTemplate in ipImageIn.

In principle you are now done with the Template Matching. By way of example, access the individual matches in the following in order to paint them into the image. For this purpose, use the construct for accessing container elements as explained in the section Container [143].

hr := F_VN_GetImageHeight(ipImageTemplate, nHeight, hr);
hr := F_VN_GetImageWidth(ipImageTemplate, nWidth, hr);
hr := F_VN_GetAt_TcVnPoint2_DINT(ipMatches, aPosition, 0, hr);
hr := F_VN_GetForwardIterator(ipMatches, ipIterator, hr);
IF SUCCEEDED(hr) AND ipIterator <> 0 THEN
    hr := ipIterator.TcQueryInterface(IID_ITcVnAccess_TcVnPoint2_DINT, ADR(ipAccess));
    IF SUCCEEDED(hr) AND ipAccess <> 0 THEN
        WHILE SUCCEEDED(hr) AND ipIterator.CheckIfEnd() <> S_OK DO
            hr := ipAccess.Get(aPosition);
            hr := F_VN_DrawRectangle(DINT_TO_UDINT(aPosition[0]),
                                      DINT_TO_UDINT(aPosition[1]),
                                      DINT_TO_UDINT(aPosition[1])+nWidth,
                                      DINT_TO_UDINT(aPosition[1])+nHeight,
                                      ipImageRes,
                                      aGreen,
                                      5,
                                      hr);
            hr := F_VN_IncrementIterator(ipIterator, hr);
        END_WHILE
    END_IF
END_IF

NOTE

Long execution duration

Depending on the size of the input and template image, the performance of the Template Matching can take a very long time in comparison with other functions. Make sure that you set the cycle time accordingly and use a watchdog if necessary!
8.1.5.2 Average intensity in shapes of any kind

In this sample the average intensity of an image is calculated within a manually defined area of an image. For this purpose a circular image mask [148] is used.

The following functions are mainly used:

- F_VN_ImageAverageExp [1067]
- F_VN_DrawCircle [884]

Explanation

The sample Statistical image features [1407] shows how statistical image features (such as average, maximum and minimum) can be calculated. Frequently, however, these features are not calculated for the entire image, but only for certain image areas. The setting of an ROI [149] is helpful if these areas are rectangular. You have to work with image masks [148] in order to restrict the calculation to areas of any shape. These are essentially images whose intensity values describe which pixels in the original image are to be considered. A mask can thus assume any shape; this sample shows a circular mask.

For this sample a quasi-binary image with a collection of screws will be observed. One conceivable use case would be, for example, that the number of screws is to be approximately determined on the basis of the average intensity value.
Applicaton

By varying the parameters nCircleShrink and aCenterOffset, you can change the size and position of the circular mask. According to this mask, the average intensity value changes: if the mask becomes smaller and shows less of the white background, the average value decreases. If the mask shifts and shows more of the white background, the average value increases.

![Result image](image1)

Average: 215.18

![Mask image](image2)

Average: 175.83

![Result image](image3)

Average: 239.04

Other images can be loaded into this sample if necessary. Note that the mask parameters may then have to be adjusted.

Code

The sample image is loaded via a FileSource state machine as ipImageWork and processed with the following code.

First of all the properties of the mask are defined. As the mask is circular, it is described by the center point aCenter and the radius nRadius. The image center point corrected by the user-adjustable offset aCenterOffset is used as the center point of the circle. The radius is calculated as the distance from the center point to the nearest border of the image, minus the value nCircleShrink.

```c
hr := F_VN_GetImageWidth(ipImageWork, nWidth, hr);
hr := F_VN_GetImageHeight(ipImageWork, nHeight, hr);
aCenter[0] := (nWidth / 2) + aCenterOffset[0];
aCenter[1] := (nHeight / 2) + aCenterOffset[1];
nRadius := (MIN(nWidth, nHeight) / 2) - nCircleShrink;
```

A new image with the same size as the input image ipImageWork is created as the mask and painted with a white circle on a black background. The circle contains the calculated properties. The value nThickness is selected as the width -1 of the border of the circle so that the circle is filled out.

```c
hr := F_VN_CreateImageAndSetPixels(ipImageMask, nWidth, nHeight, TCVN_ET_USINT, 1, aColorBlack, hr);
hr := F_VN_DrawCircle(
    nCenterX := aCenter[0],
    ....
);```
The result is that a mask image `ipImageMask` is now available with the desired circular shape:

```

```

The average intensity value of the input image `ipImageWork` in the defined circular area is now calculated with the help of this mask `ipImageMask`. The function `F_VN_ImageAverageExp` returns the array `aAverage` of the type `TcVnVector4_LREAL` in order to cover the maximum number of image channels. As the input image in this sample only has one channel, the first array element is the desired average value and is saved separately.

```

```

```

```

Finally, the mask circle and the calculated average value are drawn in a result image.

```

```

The PLC library `Tc2_Utilities` is required for the function `LREAL_TO_FMTSTR`.

### 8.1.5.3 Statistical image features

In this sample, statistical features (min, max, median, average value, standard deviation) of images are calculated. The following functions are used for this:

- `F_VN_MinPixelValue`
- `F_VN_MaxPixelValue`
- `F_VN_ImageMedian`
- `F_VN_ImageAverageStdDev`

**Application**

This project contains a File Source into which any RGB images can be loaded. These are converted in the program into grayscale images and analyzed with regard to various statistics. The result values are drawn in the image and can be viewed in the ADS Image Watch under `ipImageRes`. 
### Program

```c
// Prepare images
hr := F_VN_ConvertColorSpace(ipImageIn, ipImageIn, TCVN_CST_RGB_TO_GRAY, hr);
hr := F_VN_CopyImage(ipImageIn, ipImageRes, hr);

// Calculate image statistics
hr := F_VN_MinPixelValue(
    ipImage := ipImageIn,
    aMinValue := aMin,
    aPosition := aPos,
    hrPrev := hr
);
hr := F_VN_MaxPixelValue(
    ipImage := ipImageIn,
    aMaxValue := aMax,
    aPosition := aPos,
    hrPrev := hr
);
hr := F_VN_ImageMedian(
    ipSrcImage := ipImageIn,
    aMedian := aMedian,
    hrPrev := hr
);
hr := F_VN_ImageAverageStdDev(
    ipSrcImage := ipImageIn,
    aAverage := aAverage,
    aStdDev := aStdDev,
    hrPrev := hr
);

// Draw results
sText := CONCAT('Min: ', LREAL_TO_FMTSTR(aMin[0], 2, TRUE ));
hr := F_VN_PutLabel(sText, ipImageRes, nTextX, (nTextYBase + (1 * nTextYIncrement)), fFontScale, hr);
sText := CONCAT('Max: ', LREAL_TO_FMTSTR(aMax[0], 2, TRUE ));
hr := F_VN_PutLabel(sText, ipImageRes, nTextX, (nTextYBase + (2 * nTextYIncrement)), fFontScale, hr);
sText := CONCAT('Median: ', LREAL_TO_FMTSTR(aMedian[0], 2, TRUE ));
hr := F_VN_PutLabel(sText, ipImageRes, nTextX, (nTextYBase + (3 * nTextYIncrement)), fFontScale, hr);
sText := CONCAT('Average: ', LREAL_TO_FMTSTR(aAverage[0], 2, TRUE ));
hr := F_VN_PutLabel(sText, ipImageRes, nTextX, (nTextYBase + (4 * nTextYIncrement)), fFontScale, hr);
sText := CONCAT('StdDev: ', LREAL_TO_FMTSTR(aStdDev[0], 2, TRUE ));
hr := F_VN_PutLabel(sText, ipImageRes, nTextX, (nTextYBase + (5 * nTextYIncrement)), fFontScale, hr);
hr := F_VN_TransformIntoDisplayableImage(ipImageIn, ipImageInDisp, hr);
hr := F_VN_TransformIntoDisplayableImage(ipImageRes, ipImageResDisp, hr);
```

The PLC library Tc2_Utilities is required for the function LREAL_TO_FMTSTR.

### 8.1.5.4 Inner circle by means of distance transformation

In this sample you will find the inner circle of a misshapen object by means of distance transformation. For this purpose you essentially use the following functions:

- `F_VN_DistanceTransformation` [1063]
- `F_VN_DrawCircle` [884]

#### Explanation

External contour approximations such as an enclosing rectangle or the convex envelope can be calculated via appropriate API functions. Conversely, an inner circle with a maximum size inside a contour can be found by means of a distance transformation.
For this purpose, the point with the greatest Euclidean distance to the object edges is determined with a distance transformation. This point is then the center point of the inner circle of the object.

If you use this sample with your own images, you should adjust the threshold value and the binarization via the variables \texttt{bInvert} and \texttt{fThreshold}. By means of the parameters \texttt{eDistanceType} and \texttt{eMaskSize} you can purposefully change the parameters of the function \texttt{F\_VN\_DistanceTransformation [\textcolor{red}{1063}].}

**Program**

The first step is to binarize the input image. This achieves a clear distinction between object and background. The binarization is important for the distance transformation, as the distance transformation only reacts to pixels with the value 0. If the object is black on a white background, it must be inverted by means of the threshold type \texttt{BINARY\_INV}.

\begin{verbatim}
IF bInvert THEN
  eThresholdType := TCVN\_TT\_BINARY\_INV;
ELSE
  eThresholdType := TCVN\_TT\_BINARY;
END_IF
hr := F\_VN\_Threshold(ipImageIn, ipImageWork, fThreshold, 255, eThresholdType, hr);
\end{verbatim}

Through the distance transformation you obtain an image in which each pixel value indicates the distance to the next pixel with a value of 0. The pixel with the highest value is thus the pixel with the greatest distance. If you use the L2 norm (Euclidean distance) as the distance type \texttt{eDistanceType}, you could interpret this pixel as the center point of a circle that fills out the object as much as possible. Therefore, determine the pixel with the maximum value using the function \texttt{F\_VN\_MaxPixelValue [\textcolor{red}{1075}]} and use its position as the center point of the circle and its value as the radius of the circle.

\begin{verbatim}
hr := F\_VN\_DistanceTransformation;
    ipSrcImage := ipImageWork, 
    ipDestImage := ipImageWork, 
    eDistanceType := TCVN\_DT\_L2, 
    eMaskSize := TCVN\_DTM\_5, 
    hrPrev := hr
\end{verbatim}
The mask size eMaskSize determines the accuracy of the distance calculation and has little influence on the result.

Finally, paint the circle in the result image on the basis of the determined center point and radius in order to be able to validate the execution.

```
hr := F_VN_DrawCircle(
    nCenterX := TO_UDINT(aCenter[0]),
    nCenterY := TO_UDINT(aCenter[1]),
    nRadius := TO_UDINT(fRadius),
    ipDestImage := ipImageRes,
    aColor := aRed,
    nThickness := 2,
    hrPrev := hr);
hr := F_VN_DrawPointExp(TO_UDINT(aCenter[0]), TO_UDINT(aCenter[1]), ipImageRes, TCVN_DS_X, aRed, 10, 2, TCVN_LT_ANTIALIASED, hr);
```

## 8.1.6 Image Color and Contrast Processing

### 8.1.6.1 Color similarity with RGB reference color

In this sample, the function `F_VN_ReferenceColorSimilarityExp_TcVnVector3_LREAL` calculates the similarity of each pixel to 3 reference colors and returns them as grayscale images. The objects in the reference colors are then detected on the basis of these grayscale images.

**Variables**

- `hr` : HRESULT;
- `ipImageIn` : ITCvnImage;
- `ipImageInDisp` : ITCvnDisplayableImage;
- `ipImageThres` : ITCvnImage;
- `ipImageThresDisp` : ARRAY [0..2] OF ITCvnDisplayableImage;
- `ipImageWorkCol` : ITCvnImage;
- `ipImageWorkColDisp` : ARRAY [0..2] OF ITCvnDisplayableImage;
- `ipImageRes` : ITCvnImage;
- `ipImageResDisp` : ITCvnDisplayableImage;
- `IColor` : INT;
- `aColorTxt` : ARRAY [0..2] OF STRING := ['RED', 'GREEN', 'BLUE'];
- `aColor` : ARRAY [0..2] OF TcVnVector4_LREAL := [[150, 0, 0], [0, 255, 0], [0, 0, 255]];
- `aColorRef` : ARRAY [0..2] OF TcVnVector3_LREAL := [[255, 75, 60], [40, 140, 95], [40, 140, 190]];
- `ipContourList` : ITCvnContainer;
- `ipIterator` : ITCvnForwardIterator;
- `aOffset` : TcVnPoint;
- `ipContour` : ITCvnContainer;
- `fArea` : LREAL;
- `aCenter` : TcVnPoint2_LREAL;

**Code**

```
// Attention: With other images another color space transformation could be necessary
hr := F_VN_ConvertColorSpace(ipImageIn, ipImageRes, TCVN_CST_Bayer_RG_TO_RGB, hr);
FOR iColor := 0 TO 2 DO
    // Compute the Color Similarity to a Reference Color
    hr := F_VN_ReferenceColorSimilarityExp_TcVnVector3_LREAL(..........);  // continued
Samples

The input image in Bayer RG format

After converting the input image to an RGB image with the function `F_VN_ConvertColorSpace` [1083], the `F_VN_ReferenceColorSimilarityExp_TcVnVector3_LREAL` [1101] can be used to determine the color similarity of each pixel compared to the reference color. The following applies in the returned result image:
- The pixel value 255 (white) corresponds to a 100% match with the reference color
- The pixel value 0 (black) corresponds to a 0% match with the reference color
- and the pixel values between 0 and 255 a partial correlation to the reference color.

```c
// Compute the Color Similarity to a Reference Color
hr := F_VN_ReferenceColorSimilarityExp_TcVnVector3_LREAL(
    ipSrcImage := ipImageRes,
    ipDestImage := ipImageWorkCol,
    aRefColor := aColorRef[iColor],
    fVariance := 0.1,
    fLuminanceWeight := 0.2,
    hrPrev := hr);
```

A threshold value (F_VN_Threshold [1166]) is used to define how close the match to the reference color should be in order to be assigned to it.

```c
hr := F_VN_Threshold(ipImageWorkCol, ipImageThres, 200, 255, TCVN_TT_Binary, hr);
```

The result image, after subsequent detection.
**Similar sample**

- Check Color Range with RGB range [› 1420]

### 8.1.6.2 Display with color tables

In this sample you apply color tables to the grayscale images. You essentially use the following functions:

- F.VN.GenerateColorMap [› 1084]
- F.VN.CreateContainerFromArray [› 694]
- F.VN.GenerateCustomColorMap [› 1086]
- F.VN.ApplyColorMap [› 1079]

**Explanation**

There are several reasons why you would want to apply a color table to a monochrome image. These include, for example, a more pleasant appearance and a display of the value range that is clearer to the human eye. This is above all relevant for 16-bit images, as the value spectrum is significantly more recognizable if it is distributed over three color channels.

Depending on the use case, you would like to use various color tables. TwinCAT Vision provides some predefined color gradients via the enum ETcVnColorMap [› 160]. You can also define your own color gradients and create color tables with them.

In this sample you apply various color tables to the following grayscale gradient (with a value range of 16 bits).
Of course, you can also use your own images. Note that you can only use 1-channel images, or you will have to perform an appropriate conversion beforehand using `F_VN_ConvertColorSpace`.

### Program

Before a color table can be applied to an image, it must first be created. To this end, it must be known whether the color table is to be for 8-bit or 16-bit images, as depending on that it will require a different number of entries (see Color tables). Accordingly, you must set the parameter `eColorMapSize` of the type `ETcVnColorMapSize` to 256 in the case of 8-bit and 65536 in the case of 16-bit to match the bit depth of the input image `ipImageIn`. You achieve this by querying the element size of the pixel format by means of `F_VN_GetImageInfo`:

```pascal
hr := F_VN_GetImageInfo(ipImageIn, stImageInfo, hr);
IF stImageInfo.stPixelFormat.nElementSize = 8 THEN
    eColorMapSize := TCVN_CMS_256;
ELSIF stImageInfo.stPixelFormat.nElementSize = 16 THEN
    eColorMapSize := TCVN_CMS_65536;
END_IF
```

A color table is generated on the basis of a color gradient. You can either manually define a color gradient or use one from a template. This sample deals with the distinction on the basis of the parameter `bUseCustomColors`.

```pascal
IF bUseCustomColors THEN
    // create color map from custom-defined colors
    ...
ELSE
    // create color map from preset
    ...
END_IF
```

You define a color gradient manually by creating a container with several entries of the type `TcVnVector3_REAL`. Each entry represents a color. The colors are interpreted as being evenly distributed over the definition area when creating the color table; all other values are linearly interpolated for each of the three channels. A color gradient from the array `aColors` with the three colors red, green and blue is used here. You then create the color table with the function `F_VN_GenerateCustomColorMap`.

```pascal
VAR
    aColors: ARRAY [0..(N_COLORS-1)] OF TcVnVector3_REAL := [
        [255, 0, 0],
        [0, 255, 0],
        [0, 0, 255]
    ];
END_VAR

hr := F_VN_CreateContainerFromArray(ADR(aColors), ipColors, ContainerType_Vector_TcVnVector3_REAL, N_COLORS, hr);
hr := F_VN_GenerateCustomColorMap(
    ipColorMap := ipColorMap,
    ipInitialColors := ipColors,
    eColorMapSize := eColorMapSize,
    hrPrev := hr
);
```

In order to use a predefined color gradient, use the function `F_VN_GenerateColorMap` and set the parameter `eColorMap` to the desired color gradient from the enum `ETcVnColorMap`. In this case this is initially set to `JET`. 
VAR
  eColorMap: ETcVnColorMap := TCVN_CM_JET;
END_VAR

hr := F_VN_GenerateColorMap(
  ipColorMap := ipColorMap,
  eColorMap := eColorMap,
  eColorMapSize := eColorMapSize,
  hrPrev := hr
);

The color table created is a container that stores a color to be issued for every possible value in the input image. Therefore you can also create a color table yourself by defining all 256 or 65536 entries and then directly creating a container out of them. In most cases, however, the use of the API functions is simpler.

Finally, apply the color table created to the input image to obtain the desired result. The result image is always a 3-channel 8-bit image.

hr := F_VN_ApplyColorMap(
  ipSrcImage := ipImage,
  ipDestImage := ipImage,
  ipColorMap := ipColorMap,
  hrPrev := hr
);

The application of a predefined JET color table to the input image shown above produces the following result:

![Color gradient as result of the application of the JET color table](image)

Fig. 35: Color gradient as result of the application of the JET color table

### 8.1.7 Image filtering

#### 8.1.7.1 Blur filter

In this sample the following blur filters are compared:

- **Bilateral filter**
- **Gaussian filter**
- **Median filter**
- **Mean filter implemented as custom filter**

**Explanation**

In some cases an image is to be smoothed or freed from noise. For tasks such as Code Reading it is necessary at the same time to obtain a good contrast and good edges on the image so that the downstream algorithms can successfully work on it.

The **bilateral filter** is particularly suitable for the combination of image smoothing with the simultaneous retention of edges. For each pixel intensity it calculates a weighted average value of the surrounding pixels. The weighting depends not only on how far away the pixel is, but also on the difference in intensity. As a result, sharp edges are retained.
The **Gaussian filter** smooths images with the help of a two-dimensional discrete approximation to the Gaussian bell function. This allows the image noise to be reduced. Smaller structures are lost, but coarser structures on the other hand are retained. Edges are therefore less blurred than with the two following filters, but larger disturbances are not removed so well.

The **median filter** sorts all pixel values of the specified mask in ascending order and then selects the median value, to which the corresponding pixel is then set. This filter is well suited for suppressing structures (disturbances) that are smaller than the applied filter mask and is very robust towards outliers. As a result, the small salt-and-pepper disturbances, for example, are removed best, but the disadvantage of this is that the other detailed information is also lost.

The **custom filter** function was used in order to form a mean value. Here, the value of each pixel is replaced by the mean value of the kernel pixel, which results in blurring. Disturbances (noise) are suppressed as a result, but details and edges gradually become blurred as the number of kernels increases. As the custom filter function offers many different options for applying your own filters, the `F_VN_InitMatrixStruct` function is additionally required for the standardization of the filter kernel.

**Variables**

```plaintext
// Bilateral Filter
ipImageBilateral : ITcVnImage;
ipImageBilateralDisp : ITcVnDisplayableImage;
nBilateral_Diameter : DINT := 7;
fBilateral_SigmaColor : LREAL := 100;
fBilateral_SigmaSpace : LREAL := 100;

// Gaussian Filter
ipImageGaussian : ITcVnImage;
ipImageGaussianDisp : ITcVnDisplayableImage;
nFilterWidth : UDINT := 7;
nFilterHeight : UDINT := 7;

// Median Filter
ipImageMedian : ITcVnImage;
ipImageMedianDisp : ITcVnDisplayableImage;
nMedian_FilterSize : UDINT := 7;

// Custom Filter e.g. Mean
ipImageCustom : ITcVnImage;
ipImageCustomDisp : ITcVnDisplayableImage;
stKernelMatrix : TcVnMatrix;
// 7x7 Mean Filter Kernel with weights of 1/49 ~ 0.0204081632653
aMatrixArray7x7 : ARRAY [0..6,0..6] OF REAL := [49(0.0204081632653)];
```

**Code**

```plaintext
hr := F_VN_BilateralFilter(  _ipSrcImage := ipImageIn,
    _ipDestImage := ipImageBilateral,
    _nDiameter := nBilateral_Diameter,
    _fSigmaColor := fBilateral_SigmaColor,
    _fSigmaSpace := fBilateral_SigmaSpace,
    _hrPrev := hr);
hr := F_VN_GaussianFilter(  _ipSrcImage := ipImageIn,
    _ipDestImage := ipImageGaussian,
    _nFilterWidth := nFilterWidth,
    _nFilterHeight := nFilterHeight,
    _hrPrev := hr);
hr := F_VN_MedianFilter(  _ipSrcImage := ipImageIn,
    _ipDestImage := ipImageMedian,
    _nFilterSize := nMedian_FilterSize,
    _hrPrev := hr);

// Mean-Filter with Custom Filter Function
hr := F_VN_InitMatrixStruct(  _pSrcBuffer := ADR(aMatrixArray7x7),
    _stDestMatrix := stKernelMatrix,
    _nRows := 7,
    _nCols := 7,
    _eElementType := TCVN_ET_REAL,
    _hrPrev := hr);
```
Result

The unprocessed original image (1st row) already exhibits fine structures on the surface of the gear wheels in the detail. In order to illustrate the effects of the filters, additional disturbances were added to the original image by Gauss noise (2nd row) and salt-and-pepper noise (3rd row).

So that the effects of the different filters are as comparable as possible, a kernel size of 7 x 7 is used with all of them. Using other parameter values or kernel sizes, as well as further setting options for the EXP variants of the functions, other effects can be achieved in detail. As the filter effect very much depends on the content of the input image, it is recommended to test several filters with different parameter settings in order to compare them directly and find the best filter for the use case. The table below contains an overview of the result images:

<table>
<thead>
<tr>
<th>Original images</th>
<th>Bilateral filter</th>
<th>Gaussian filter</th>
<th>Median filter</th>
<th>Mean filter</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Original Image" /></td>
<td><img src="image2" alt="Bilateral Filter" /></td>
<td><img src="image3" alt="Gaussian Filter" /></td>
<td><img src="image4" alt="Median Filter" /></td>
<td><img src="image5" alt="Mean Filter" /></td>
</tr>
<tr>
<td><img src="image6" alt="Original Image" /></td>
<td><img src="image7" alt="Bilateral Filter" /></td>
<td><img src="image8" alt="Gaussian Filter" /></td>
<td><img src="image9" alt="Median Filter" /></td>
<td><img src="image10" alt="Mean Filter" /></td>
</tr>
<tr>
<td><img src="image11" alt="Original Image" /></td>
<td><img src="image12" alt="Bilateral Filter" /></td>
<td><img src="image13" alt="Gaussian Filter" /></td>
<td><img src="image14" alt="Median Filter" /></td>
<td><img src="image15" alt="Mean Filter" /></td>
</tr>
</tbody>
</table>

**NOTE**

**Long or fluctuating execution times**

The duration of the execution of a filter essentially depends on the type of filter as well as the image and kernel size. Therefore, please note that a filter operation can take considerably longer than other functions, so you should adapt the cycle time of your task accordingly.

If the filter parameters are changed during the runtime, in particular the kernel size, it is recommended to use watchdogs.[137]. In the case of the median filter, the runtime can always fluctuate somewhat due to the principle.

Furthermore, job tasks [61] can lead to a reduction in the total computing time, which should be tested according to the system.

### 8.1.7.2 Edge detection filters

In this sample the following edge detection filters are compared:

- **Sobel filter [1157]**
- **Scharr filter [1149]**
- **Laplace filter [1135]**
Explanation

The Sobel filter is a simple linear filter for marking gradients in an image. The Scharr filter is very similar to this; only the characteristic of the filter kernel is somewhat different. The Scharr filter has a better isotropy, i.e. the detection of edges is independent of the direction.

The Laplace filter detects edges based on the zero points of the discrete 2nd derivation of the image. This variant is susceptible to noise by comparison.

Variables

```
// Sobel
ipImageSobel : ITCvNImage;
ipImageSobelDisp : ITCvNDisplayableImage;
eSobel_DestDepth : ETCvNElementType := TCVN_ET_USINT;
aSobel_DerivOrder : ARRAY [0..1] OF UDINT := [1, 1];
nSobel_KernelSize : UDINT := 3;
fSobel_Scale : LREAL := 10;
fSobel_Delta : LREAL := 0;
eSobel_BorderExtra : ETCvNBorderInterpolationMethod := TCVN_BIM_DEFAULT;
```

```
// Scharr
ipImageScharr : ITCvNImage;
ipImageScharrDisp : ITCvNDisplayableImage;
eScharr_DestDepth : ETCvNElementType := TCVN_ET_USINT;
eScharr_FilterDir : ETCvNFilterDirection := TCVN_FD_X;
fScharr_Scale : LREAL := 1;
fScharr_Delta : LREAL := 0;
eScharr_BorderExtra : ETCvNBorderInterpolationMethod := TCVN_BIM_DEFAULT;
```

```
// Laplace
ipImageLaplace : ITCvNImage;
ipImageLaplaceDisp : ITCvNDisplayableImage;
eLaplace_DestDepth : ETCvNElementType := TCVN_ET_USINT;
nLaplace_KernelSize : UDINT := 3;
fLaplace_Scale : LREAL := 10;
fLaplace_Delta : LREAL := 0;
eLaplace_BorderExtra : ETCvNBorderInterpolationMethod := TCVN_BIM_DEFAULT;
```

Code

```
// Execute the Sobel Filter
hr := F_VN_SobelFilterExp(
    ipSrcImage := ipImageIn,
    ipDestImage := ipImageSobel,
    eDestDepth := eSobel_DestDepth,
    nXOrder := aSobel_DerivOrder[0],
    nYOrder := aSobel_DerivOrder[1],
    nKernelSize := nSobel_KernelSize,
    fScale := fSobel_Scale,
    fDelta := fSobel_Delta,
    eBorderType := eSobel_BorderExtra,
    hrPrev := hr);
```

```
// Execute the Scharr Filter
hr := F_VN_ScharrFilterExp(
    ipSrcImage := ipImageIn,
    ipDestImage := ipImageScharr,
    eDestDepth := eScharr_DestDepth,
    eFilterDirection := eScharr_FilterDir,
    fScale := fScharr_Scale,
    fDelta := fScharr_Delta,
    eBorderType := eScharr_BorderExtra,
    hrPrev := hr);
```

```
// Execute the Laplacian Filter
hr := F_VN_LaplacianFilterExp(
    ipSrcImage := ipImageIn,
    ipDestImage := ipImageLaplace,
    eDestDepth := eLaplace_DestDepth,
    nKernelSize := nLaplace_KernelSize,
    fScale := fLaplace_Scale,
    fDelta := fLaplace_Delta,
    eBorderType := eLaplace_BorderExtra,
    hrPrev := hr);
```
Results

The sample project contains the functions calls of the three filters. The parameter configuration and its effect on a sample image can be tested in this sample.

<table>
<thead>
<tr>
<th>Original image</th>
<th>Sobel filter</th>
<th>Scharr filter</th>
<th>Laplace filter</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Original image" /></td>
<td><img src="image2.png" alt="Sobel filter" /></td>
<td><img src="image3.png" alt="Scharr filter" /></td>
<td><img src="image4.png" alt="Laplace filter" /></td>
</tr>
</tbody>
</table>

8.1.7.3 Morphological operators

In this sample you apply morphological operators to an image. For this you use:

- `F_VN_CreateStructuringElement` [1121]
- `F_VN_MorphologicalOperator` [1145]

Explanation

Objects in an image can be changed according to certain rules with a morphological operator. Morphological operators are mostly applied to binary images; however, application to grayscale images is generally also possible. Through combined operators such as OPENING and CLOSING it is possible, for example, to supplement missing parts in objects and small outliers can be removed.

Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipElement</td>
<td>ITcVnImage</td>
</tr>
<tr>
<td>eSEShape</td>
<td>ETcVnStructuringElementShape := TCVN_SES_RECTANGLE;</td>
</tr>
<tr>
<td>aSESize</td>
<td>ARRAY [0..1] OF UDINT := [29, 29];</td>
</tr>
<tr>
<td>eOperator</td>
<td>ETcVnMorphologicalOperator := TCVN_MO_OPENING;</td>
</tr>
</tbody>
</table>

Program

A morphological operator requires a structure element in order to define its area of influence for each pixel. For this purpose you can select various shapes from the enum `ETcVnStructuringElementShape` [200] and in different sizes. You then define a morphological operation by means of the enum `ETcVnMorphologicalOperator` [195] and apply it with the structure element created to the input image.

```pascal
hr := F_VN_CreateStructuringElement(
   ipStructuringElement := ipElement,
   eShape := eSEShape,
   nWidth := aSESize[0],
   nHeight := aSESize[1],
   hrPrev := hr);
```

```pascal
hr := F_VN_MorphologicalOperator(
   iP SrcImage := ipImageIn,
   iP DestImage := ipImageRes,
   eOperator := eOperator,
   ipStructuringElement := ipElement,
   hrPrev := hr);
```

A morphological opening with a rectangular structure element of the size [31; 31] and previous threshold value, for example, produces the following result. It removes smaller or narrower objects from the image.
**NOTE**

**Long execution duration**
Depending on the size of the input image and of the structure element, the execution of the morphological operator can take a very long time in comparison with other functions. Make sure that you set the cycle time accordingly and use a watchdog if necessary!

### 8.1.8 Image segmentation

- Check Color Range with RGB range [1420]

#### 8.1.8.1 Check Color Range with RGB range

In this sample, the function F_VN_CheckColorRange [1165] searches for red, green and blue objects in the image and labels them according to their color. The function is applied to RGB images. It is also possible to apply the function to images in other color spaces such as HSV, Lab and BGR, in which case the parameterization must be adapted accordingly.

**Variables**

```c
hr : HRESULT;

// Images
ipImageIn : ITCnImage;
ipImageInDisp : ITCnDisplayableImage;
ipImageWorkCol : ITCnImage;
ipImageWorkColDisp : ARRAY [0..2] OF ITCnDisplayableImage;
ipImageRes : ITCnImage;
ipImageResDisp : ITCnDisplayableImage;

// Colors
IColor : INT;
aColorTxt : ARRAY[0..2] OF STRING := ['RED', 'GREEN', 'BLUE'];
aColor : ARRAY[0..2] OF TcVnVector4_LREAL := [[150, 0, 0], [0, 255, 0], [0, 0, 255]];
aColorRefLow : ARRAY[0..2] OF TcVnVector4_LREAL := [[150, 50, 20], [35, 90, 60], [20, 40, 130]];
aColorRefUp : ARRAY[0..2] OF TcVnVector4_LREAL := [[255, 120, 100], [100, 200, 140], [60, 160, 255]];

// Contours
ipContourList : ITCnContainer;
ipIterator : ITCnForwardIterator;
ipContour : ITCnContainer;
fArea : LRREAL;
aCenter : TcVnPoint2_LREAL;
```
Code

// Attention: With other images another color space transformation could be necessary
hr := F_VN_ConvertColorSpace(ipImageIn, ipImageRes, TCVN_CST_Bayer_RG_TO_RGB, hr);

FOR iColor := 0 TO 2 DO

    // Apply a "Color-Threshold" on the image
    hr := F_VN_CheckColorRange(ipSrcImage := ipImageRes,
                               ipDestImage := ipImageWorkCol,
                               aLowerBounds := aColorRefLow[iColor],
                               aUpperBounds := aColorRefUp[iColor],
                               hrPrev := hr);

    // Find all objects / contours in the black and white image
    hr := F_VN_FindContours(ipImageWorkCol, ipContourList, hr);
    hr := F_VN_GetForwardIterator(ipContourList, ipIterator, hr);

    // Filter the objects by size and draw the contours
    WHILE SUCCEEDED(hr) AND_THEN ipIterator.CheckIfEnd() <> S_OK DO
        hr := F_VN_GetContainer(ipIterator, ipContour, hr);
        hr := F_VN_IncrementIterator(ipIterator, hr);
        // Filter contours by size
        hr := F_VN_ContourArea(ipContour, fArea, hr);
        IF fArea > 5000 THEN
            // Draw Results into an Image
            hr := F_VN_DrawContours(ipContour, -1, ipImageRes, aColor[iColor], 3, hr);
            hr := F_VN_ContourCenterOfMass(ipContour, aCenter, hr);
            hr := F_VN_PutText(aColorTxt[iColor], ipImageRes, LREAL_TO_UDINT(aCenter[0])-30,
                                LREAL_TO_UDINT(aCenter[1])+10, TCVN_FT_HERSHEY_PLAIN, 2, aColor[iColor], hr);
        END_IF
    END_WHILE

END_FOR

Result

The input image in Bayer RG format.
The input image is converted into an RGB image with the function `F_VN_ConvertColorSpace`.

```c
hr := F_VN_ConvertColorSpace(ipImageIn, ipImageRes, TCVN_CST_Bayer_RG_TO_RGB, hr);
```

The function `F_VN_CheckColorRange` can then be used with the respective upper and lower threshold values.

```c
hr := F_VN_CheckColorRange(ipImageRes, ipImageWorkCol, aColorRefLow[iColor], aColorRefUp[iColor], hr);
```

<table>
<thead>
<tr>
<th>Red</th>
<th>Green</th>
<th>Blue</th>
</tr>
</thead>
<tbody>
<tr>
<td>aColorRefLow := [150,50,20];</td>
<td>aColorRefLow := [35,90,60];</td>
<td>aColorRefLow := [20,40,130];</td>
</tr>
<tr>
<td>aColorRefUp := [255,120,100];</td>
<td>aColorRefUp := [100,200,140];</td>
<td>aColorRefUp := [60,160,255];</td>
</tr>
</tbody>
</table>

The result image, after subsequent detection
8.1.9 Measurement

8.1.9.1 Locate Circular Arc

In this example, the function F_VN_LocateCircularArc is used to locate and measure circular arcs. This function returns a TcVnCircularArc structure containing the center, radius and bounding angles as result. To measure complete circles, use the function F_VN_LocateEllipse instead.

The start position of the search should correspond approximately to the expected circle center of the circular arc, which is 400, 300 in the corresponding sample images.

To demonstrate that an approximate position is sufficient here, the corresponding parameter aCenter in the sample was set to 420, 310. The sample illustrates that it is possible to compensate small positional changes of the objects to be examined in the image.

Since the actual radius in the sample images is about 200 pixels and the starting point is deliberately inaccurate, the search radius fRadius was set to 270. In the specific sample, around 230 pixels would have been sufficient, but in practice it is always good to have an additional safety buffer, e.g. if the position of the object deviates more than expected.

For the sample images, the search direction must be selected so that the objects are found correctly in all images. A good direction would be down left, around 2,356 rad (135°). To illustrate that this direction does not have to be exact, fDirection was set to 2.1 rad (120.3°) in the sample.
In addition, a transition from light to dark, starting from the starting point, should be specified as a search criterion. The minimum contrast also has to be specified, above which the function detects an intensity difference between adjacent pixels as an edge.

In case these parameters are not sufficient to achieve good results, the expert function `F_VN_LocateCircularArcExp` offers a number of additional parameters, which are not considered in this sample.

**Variables**

```pascal
hr : HRESULT;
ipImageIn : ITcVnImage;
ipImageInDisp : ITcVnDisplayableImage;
ipImageRes : ITcVnImage;
ipImageResDisp : ITcVnDisplayableImage;

// result
stArc : TcVnCircularArc;

// input parameters (to specify where to start searching for the circular arc)
aCenter : TcVnPoint2_REAL := [420, 310];
fRadius : REAL := 270;
fDirection : LREAL := 2.1;

// drawing
aColor : TcVnVector4_LREAL := [0, 175, 0];
sText : STRING(255);
```

**Code**

```pascal
hr := F_VN_LocateCircularArc(
ipSrcImage := ipImageIn,
stCircularArc := stArc,
aCenterPoint := aCenter,
fSearchRadius := fRadius,
fArcDirectionRad := fDirection,
eEdgeDirection := TCVN_ED_LIGHT_TO_DARK,
fMinStrength := 100,
hrPrev := hr);

// Draw result for visualization
hr := F_VN_ConvertColorSpace(ipImageIn, ipImageRes, TCVN_CST_GRAY_TO_RGB, hr);
hr := F_VN_DrawCircularArc(stArc, ipImageRes, aColor, 2, hr);
sText := CONCAT(CONCAT(CONCAT('Center ', REAL_TO_STRING(stArc.aCenter[0])), ', '),
REAL_TO_STRING(stArc.aCenter[1]));
hr := F_VN_PutText(sText, ipImageRes, 420, 50, TCVN_FT_HERSHEY_SIMPLEX, 0.7, aColor, hr);
sText := CONCAT('Radius ', REAL_TO_STRING(stArc.fRadius));
hr := F_VN_PutText(sText, ipImageRes, 420, 75, TCVN_FT_HERSHEY_SIMPLEX, 0.7, aColor, hr);
sText := CONCAT('Angle start ', REAL_TO_STRING(stArc.fStartAngle * 180 / LREAL_TO_REAL(PI)));
hr := F_VN_PutText(sText, ipImageRes, 420, 100, TCVN_FT_HERSHEY_SIMPLEX, 0.7, aColor, hr);

// Display source and result image
hr := F_VN_TransformIntoDisplayableImage(ipImageIn, ipImageInDisp, S_OK);
hr := F_VN_TransformIntoDisplayableImage(ipImageRes, ipImageResDisp, S_OK);
```

**Results**

For visualization, the circular arc that is found is drawn into the result image using the function `F_VN_DrawCircularArc`. The underlying numerical values are also displayed as text.
Fig. 36: LocateCircularArc_3QuarterCircle_Result
Fig. 37: LocateCircularArc_HalfCircle_Result

<table>
<thead>
<tr>
<th>Center</th>
<th>399.9377, 300.2029</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radius</td>
<td>200.6076</td>
</tr>
<tr>
<td>Angle start</td>
<td>0.2947669</td>
</tr>
<tr>
<td>Angle end</td>
<td>179.7752</td>
</tr>
</tbody>
</table>
Fig. 38: LocateCircularArc_QuarterCircle_Result

**Similar samples**
- Locate Edge [1427]
- Locate Ellipse [1432]
- Measure Angle Between Edges [1437]

### 8.1.9.2 Locate Edge

In this sample,
- the function `F_VN_LocateEdgeExp` [1233] is used to locate an edge in a defined search window,
- and the execution time is monitored with a watchdog and limited if necessary.

**Explanation**

A search window can be used to locate edges with the function `F_VN_LocateEdge` [1230]. Some parameters are fixed for simplicity. Alternatively, the function `F_VN_LocateEdgeExp` [1233] that is used here provides full access to all parameters. This sample is intended to illustrate the individual parameters. We recommend trying out configuration changes in addition to the standard configuration and consider the effects on the edge result and on the processing time. A selection of configuration changes including descriptions can be found under Results.

**Variables**

```plaintext
hr : HRESULT;
hrFunc : HRESULT;
```
ipImageIn : ITcVnImage;
ipImageInDisp : ITcVnDisplayableImage;
ipImageRes : ITcVnImage;
ipImageResDisp : ITcVnDisplayableImage;

// result
ipEdgePoints : ITcVnContainer;

// parameters
aStartPoint : TcVnPoint2_REAL := [850, 400];
aEndPoint : TcVnPoint2_REAL := [550, 400];
eDirection : ETcVnEdgeDirection := TCVN_ED_DARK_TO_LIGHT;
fMinStrength : REAL := 50;
nSearchLines : UDINT := 31;
fSearchLineDist : REAL := 1;
nMaxThickness : UDINT := 7;
nSubpixIter : UDINT := 10;
eAlgorithm : ETcVnEdgeDetectionAlgorithm := TCVN_EDA_INTERPOLATION;
fAvgStrength : REAL;

// Watchdog
hrWD := HRESULT;
tStop : DINT := 15000;
tRest := DINT;
nFraction := UDINT;

// drawing
aLine : TcVnVector4_LREAL;
aColorGreen : TcVnVector4_LREAL := [0, 175, 0];
aColorBlue : TcVnVector4_LREAL := [0, 0, 255];
aColorRed : TcVnVector4_LREAL := [255, 0, 0];
sText : STRING(255);

Code

hrWD := F_VN_StartRelWatchdog(tStop, hr);
hrFunc := F_VN_LocateEdgeExp(ipSrcImage, ipEdgePoints, aStartPoint, aEndPoint, eEdgeDirection, fMinStrength, nSearchLines, fSearchLineDist, nMaxThickness, nSubpixIter, fApproxPrecision, eAlgorithm, hrPrev, fAvgStrength);

// Display source and result image
hr := F_VN_TransformIntoDisplayableImage(ipImageIn, S_OK);
h := F_VN_TransformIntoDisplayableImage(ipImageRes, S_OK);
Results

For visualization, the edge points that are found are shown in blue, and a regression line is drawn in green in the output image. In addition, the starting point (red circle) and the end point (red X) are drawn. The processing percentage, the required execution time and the return value of the function are displayed in the top left corner.

For the parameters used in this sample, the result looks like this:

```
Processed  100%
Time       384us
Returncode 0
```

If the number of search lines is increased from 31 to 61 with the parameter `nSearchLines`, the required computing time is also doubled. In return, the regression line becomes more precise:
If you want to find the milling edge of the component instead of the outer edge, change the parameter `eDirection` to `TCVN_ED_LIGHT_TO_DARK`. This ignores the outer edge transition from dark to light and instead finds the following edge with the transition from light to dark:
Reducing execution time

It is noticeable that the execution time, despite otherwise identical parameters, is higher than when locating the outer edge. This is because the edge that is found is further away from the starting point.

Conversely, this means that the execution time for a fixed search window varies slightly depending on the position of the component in the image, which can also be seen in the three sample images.

An even greater fluctuation can occur with the two approximation algorithms, as they require different iterations to approximate the model parameters using the surrounding pixel intensities.

In general, fewer iterations are required if the search lines of the search window hit the edge as orthogonally as possible. If the position and orientation of the objects in the image are subject to little or no fluctuation, the search window can be adjusted accordingly to achieve shorter execution times. In addition, the maximum time required can be reduced by reducing the maximum number of iterations, but this may lead to less accurate results.

Using a watchdog

In addition, the time can be limited with an external watchdog, which terminates the execution of the function if necessary and returns the existing partial results. For example, if the algorithm is changed to TCVN_EDA_APPROX_ERF and the maximum number of iterations is set to 60, the execution for the image LocateEdge2.bmp (outer edge orthogonal to the search lines) will take almost exactly 4 ms (depending on the installed CPU), while the other two images will take 4.3 and 4.4 ms. If the parameter tStop is set to 4000, the execution of the function is stopped as soon as possible after 4 ms, and the partial results available up to then are returned. In other words, the function on the image LocateEdge2.bmp continues to be processed 100%, whereas the function on the other two images (in this case LocateEdge3.bmp) is terminated prematurely:
Since it is only possible to abort at certain points in the algorithm, and the partial results (previously found edge points on the search lines) are returned, the function is not terminated after exactly 4000 μs but takes a little longer, which must be taken into account when selecting the abort time. The maximum additional time required generally depends on the algorithm and the parameterization.

Despite the termination, the partial results are sufficient to achieve the desired result, since only the results of one or two search lines are missing.

**Similar samples**

- Locate Circular Arc [1423]
- Locate Ellipse [1432]
- Measure Angle Between Edges [1437]

### 8.1.9.3 Locate Ellipse

In this sample,

- the function F\_VN\_LocateEllipseExp [1244] is used to locate a circular object in a defined search area
- and the execution time is monitored with a watchdog and limited if necessary.

**Explanation**

For a basic understanding of the measurement functions and their parameters, please follow the Locate Edge sample [1427] first. The function F\_VN\_LocateEllipseExp [1244] differs from F\_VN\_LocateEdgeExp [1230] in that it uses a circular search window instead of a rectangular one. In this sample a large, inaccurate search window is parameterized, since the position of the circular objects changes slightly. The circles can be determined with the same search window in all images.
The parameter `bInvertSearchDirection` which was set to `TRUE` in this sample, was used to define that the search direction is reversed, i.e. from the outside towards `aCenter`. This is necessary because the sample images also contain edges within the ellipse the system is looking for.

The search direction was set to `TCVN_ED_LIGHT_TO_DARK`, since in this sample the objects are darker than the background.

The parameter `nMaxThickness`, which defines the number of pixels with which `fMinStrength` must be reached, was set to "7".

Variables

```
hr : HRESULT;
hrFunc : HRESULT;
ipImageIn : ITcVnImage;
ipImageInDisp : ITcVnDisplayableImage;
ipImageRes : ITcVnImage;
ipImageResDisp : ITcVnDisplayableImage;
// result
stEllipse : TcVnRotatedRectangle;
ipContourPoints : ITcVnContainer;
// input parameters
aCenter : TcVnPoint2_REAL := [650, 400];
fRadius : REAL := 300;
fMinRadius : REAL := 100;
fMinStrength : REAL := 30;
nSubpixIter : UDINT := 10;
nSearchLines : UDINT := 40;
eAlgorithm : ETcVnEdgeDetectionAlgorithm := TCVN_EDA_INTERPOLATION;
// Watchdog
hrWD : HRESULT;
tStop : DINT := 15000;
tRest : DINT;
nFraction : UDINT;
// drawing
aColorRed : TcVnVector4_LREAL := [200, 0, 0];
aColorGreen : TcVnVector4_LREAL := [0, 175, 0];
sText : STRING(255);
```

Code

```
hrWD := F_VN_StartRelWatchdog(tStop, hr);
hrFunc := F_VN_LocateEllipseExp(ipSrcImage := ipImageIn,
stEllipse := stEllipse,
aCenterPoint := aCenter,
fSearchRadius := fRadius,
eEdgeDirection := TCVN_ED_LIGHT_TO_DARK,
fMinStrength := fMinStrength,
nMaxThickness := 7,
bInvertSearchDirection := TRUE,
fMinSearchRadius := fMinRadius,
nSubpixelsIterations := nSubpixIter,
nSearchLines := nSearchLines,
fApproxPrecision := 0.0001,
eAlgorithm := eAlgorithm,
ipContourPoints := ipContourPoints,
hrPrev := hr);
hrWD := F_VN_StopWatchdog(hrWD, nFractionProcessed=>nFraction, tRest=>tRest);
```

// Draw result for visualization
```
hr := F_VN_ConvertColorSpace(ipImageIn, ipImageRes, TCVN_CST_GRAY_TO_RGB, hr);
sText := CONCAT(CONCAT('Processed ', UDINT_TO_STRING(nFraction)), '%');
hr := F_VN_PutTextExp(sText, ipImageRes, 25, 50, TCVN_FT_HERSHEY_SIMPLEX, 1.3, aColorGreen, 2,
TCVN_LT_8_CONNECTED, FALSE, hr);
hr := F_VN_PutTextExp(sText, ipImageRes, 25, 100, TCVN_FT_HERSHEY_SIMPLEX, 1.3, aColorGreen, 2,
TCVN_LT_8_CONNECTED, FALSE, hr);
hr := F_VN_DrawCircle(REAL_TO_UDINT(aCenter[0]), REAL_TO_UDINT(aCenter[1]), REAL_TO_UDINT(fRadius), ipImageRes, aColorRed, 2, hr);
hr := F_VN_DrawCircle(REAL_TO_UDINT(aCenter[0]), REAL_TO_UDINT(aCenter[1]), REAL_TO_UDINT(fRadius), ipImageRes, aColorRed, 2, hr);
```
Samples

hr := F_VN_DrawPoints(ipContourPoints, ipImageRes, TCVN_DS_X, aColorGreen, hr);
hr := F_VN_DrawPoint(REAL_TO_UDINT(stEllipse.aCenter[0]), REAL_TO_UDINT(stEllipse.aCenter[1]),
ipImageRes, TCVN_DS_PLUS, aColorGreen, hr);
hr := F_VN_DrawEllipse(stEllipse, ipImageRes, aColorGreen, 1, hr);

// Display source and result image
hr := F_VN_TransformIntoDisplayableImage(ipImageIn, ipImageInDisp, S_OK);
hr := F_VN_TransformIntoDisplayableImage(ipImageRes, ipImageResDisp, S_OK);

Results

The function returns the TcVnRotatedRectangle structure stEllipse, which defines the ellipse that was found by its center aCenter, its height and width stSize.fHeight / stSize.fWidth and its angle fAngle.

Optionally, one additional edge point per search line ipContourPoints can be returned from which the ellipse was approximated.

For visualization, the search area is first drawn into the image in the form of two red circles around the specified center point. The ellipse that was found is drawn as a green line and the corresponding center as a green "+". The individual measuring points from ipContourPoints are drawn as green x. In addition, the processing percentage and the required computing time are shown at the top left. For the parameters used in this sample, the result for the input image LocateEllipse_T.bmp looks like this:

Processed 100%
Time 731us

If the number of search lines is increased from 40 to 180, the required computing time increases linearly as expected:
In practice the computing time varies somewhat, depending on the input image. In many cases it is therefore useful to limit the computing time by means of a watchdog. If, for example, the number of search lines is set to 120 and the watchdog timeout $t_{Stop}$ to 2000 (depending on the CPU used), the function is terminated at around 85% of the processing time, and the partial results available at this point are returned:
Since the algorithm can only be interrupted and the partial results returned at specific points, the function does not abort exactly after 2000 µs but takes a little longer, which must be taken into account when selecting the abort time. The maximum additional time required generally depends on the algorithm and the parameterization.

Although the results of some search lines are missing due to the termination (four relatively large gaps between the green x), the ellipse could still be correctly approximated, i.e. the partial results can be used as if the function had been processed 100%.

This is also possible to a limited extent if the position of an object is partly outside the defined search range, e.g. because a previous positioning step was incorrect. If, for example, $fMinRadius$ is set to 180, part of the outer contour of some sample images lies outside the specified range. Nevertheless, enough search points are available to approximate an ellipse. In practice, however, one should not rely on this and should set the search area large enough:
Similar samples

- Locate Circular Arc [1423]
- Locate Edge [1427]
- Measure Angle Between Edges [1437]

8.1.9.4 Measure Angle Between Edges

In this sample,

- the function `F_VN_MeasureAngleBetweenEdgesExp` is used to measure the angle between two edges of an object
- and the execution time is monitored with a watchdog and limited if necessary.

Explanation

After defining the search windows, the search direction was set to `TCVN_ED_LIGHT_TO DARK`, because in this sample the search is performed from the inside to the outside and the objects are lighter than the background.

The parameter `nMaxThickness`, which defines the number of pixels with which `fMinStrength` must be reached, was set to "7".

Variables

```plaintext
hr : HRESULT;
hrFunc : HRESULT;
ipImageIn : ITcVnImage;
ipImageInDisp : ITcVnDisplayableImage;
```
Samples

TF7000 - TF7300

1438

Version: 1.3

Code

hrWD := F_VN_StartRelWatchdog(tStop, hr);
hrFunc := F_VN_MeasureAngleBetweenEdgesExp(
    ipSrcImage := ipImageIn,
    fAngle := fAngle,
    aInnerPoint := aInnerPoint,
    aOuterPoint1 := aOuterPoint1,
    aOuterPoint2 := aOuterPoint2,
    eEdgeDirection := TCVN_ED_LIGHT_TO_DARK,
    fMinStrength := fMinStrength,
    nSearchLines := nSearchLines,
    fSearchLineDist := fSearchLineDist,
    nMaxThickness := 7,
    nSubpixIter := nSubpixIter,
    bAngleInDegrees := TRUE,
    fApproxPrecision := 0.0001,
    eAlgorithm := eAlgorithm,
    ipEdgePoints1 := ipEdgePoints1,
    ipEdgePoints2 := ipEdgePoints2,
    hrPrev := hr);
hrWD := F_VN_StopWatchdog(hrWD, nFractionProcessed=>nFraction, tRest=>tRest);

hr := F_VN_ConvertColorSpace(ipImageIn, ipImageRes, TCVN_CST_GRAY_TO_RGB, hr);
sText := CONCAT('Angle ', REAL_TO_STRING(fAngle));
hr := F_VN_PutTextExp(sText, ipImageRes, 25, 200, TCVN_FT_HERSHEY_SIMPLEX, 1.3, aColorGreen, 2,
    TCVN_LT_8_CONNECTED, FALSE,hr);
sText := CONCAT(CONCAT('Time ', DINT_TO_STRING(tStop - tRest)), 'us');
hr := F_VN_PutTextExp(sText, ipImageRes, 25, 250, TCVN_FT_HERSHEY_SIMPLEX, 1.3, aColorGreen, 2,
    TCVN_LT_8_CONNECTED, FALSE,hr);
hr := F_VN_DrawPointExp(REAL_TO_UDINT(aInnerPoint[0]), REAL_TO_UDINT(aInnerPoint[1]), ipImageRes,
    TCVN_DS_CIRCLE, aColorGreen, 3, 2, TCVN_LT_8_CONNECTED, FALSE,hr);
hr := F_VN_DrawPointExp(REAL_TO_UDINT(aOuterPoint1[0]), REAL_TO_UDINT(aOuterPoint1[1]), ipImageRes,
    TCVN_DS_CIRCLE, aColorGreen, 3, 2, TCVN_LT_8_CONNECTED, FALSE,hr);
hr := F_VN_DrawPointExp(REAL_TO_UDINT(aOuterPoint2[0]), REAL_TO_UDINT(aOuterPoint2[1]), ipImageRes,
    TCVN_DS_CIRCLE, aColorGreen, 3, 2, TCVN_LT_8_CONNECTED, FALSE,hr);
hr := F_VN_DrawLine_TcVnVector4_LREAL(aLine1, ipImageRes, aColorGreen, 2, hr);
hr := F_VN_DrawLine_TcVnVector4_LREAL(aLine2, ipImageRes, aColorYellow, 3, 2, TCVN_LT_8_CONNECTED, hr);
hr := F_VN_DrawLine_TcVnVector4_LREAL(aLine1, ipImageRes, aColorRed, 3, 2, TCVN_LT_8_CONNECTED, hr);
hr := F_VN_DrawLine_TcVnVector4_LREAL(aLine2, ipImageRes, aColorYellow, 3, 2, TCVN_LT_8_CONNECTED, hr);
hr := F_VN_DrawPointsExp(ipEdgePoints1, ipImageRes, TCVN_DS_PLUS, aColorRed, 1, 1,
    TCVN_LT_8_CONNECTED, hr);
hr := F_VN_DrawPointsExp(ipEdgePoints2, ipImageRes, TCVN_DS_PLUS, aColorYellow, 1, 1,
    TCVN_LT_8_CONNECTED, hr);
// Display source and result image
hr := F_VN_TransformIntoDisplayableImage(ipImageIn, ipImageInDisp, S_OK);
hr := F_VN_TransformIntoDisplayableImage(ipImageRes, ipImageResDisp, S_OK);

Results

For visualization purposes, aInnerPoint is first displayed as a green circle, aOuterPoint1 as a red x and aOuterPoint2 as a yellow x. The corresponding edge points ipEdgePoints1 and ipEdgePoints2 that were found are also drawn in red or yellow. The lines approximated from the edge points are drawn in green. The calculated angle fAngle in degrees and the required computing time in µs are displayed on the left in the image. For the parameters used in this sample, the result looks like this:

To get a more accurate result, eAlgorithm can be changed to TCVN_EDA_APPROX_ERF and nSubpixIter to 50. However, this also significantly increases the computing time required:
Samples

TF7000 - TF7300

Version: 1.3

8.1.10 Self-written functions

This sample explains what needs to be observed when writing your own functions for Vision applications. It essentially concerns:

- the handling of the HRESULT
- the handling of interface pointers

Other structures too

This function implementation is based on the structure of the TwinCAT Vision API functions. Other structures are also conceivable, but please pay attention to the points explained here.

Methods too

In this sample a function is described. However, it applies equally to methods.

For this purpose, by way of example, a self-written function for the counting and painting of objects is considered:

FUNCTION F_CountAndDrawObjects : HRESULT
Declaration part

The attributes of the function are declared as follows in the \texttt{VAR\_INPUT} part. The original image is transferred directly as \texttt{ITcVnImage}. Since no variable values are stored in a function or method, nothing else needs to be observed in this case. On the other hand, the result image is passed as \texttt{REFERENCE TO ITcVnDisplayableImage}, since it is created in the function and is to be returned externally for further processing. In this case, no copy of the interface pointer is created due to the reference.

\begin{verbatim}
VAR\_INPUT
  ipSrcImage   :   ITcVnImage;
  ipDestImage  :   REFERENCE TO ITcVnImage;
  aColor       :   TcVnVector4\_LREAL;
  nNumberOfObjects : REFERENCE TO ULINT;
  hrPrev       :   HRESULT;
END_VAR
\end{verbatim}

In addition, some auxiliary variables required internally by the function are declared in the \texttt{VAR}.

\begin{verbatim}
VAR
  hr          :   HRESULT;
  ipContours  :   ITcVnContainer;
  stParams    :   TcVnParamsBlobDetection;
END_VAR
\end{verbatim}

Input checks

At the start of the function, some input checks are performed in order to ensure that the function can be executed properly.

Initially, \texttt{FAILED(hrPrev)} is used to check whether an error has occurred in the previous processing chain. If so, the same error is displayed and the function is ended directly.

\begin{verbatim}
IF FAILED(hrPrev) THEN
  F\_CountAndDrawObjects := hrPrev;
  RETURN;
END_IF
\end{verbatim}

A check is also performed to ascertain whether all interface pointers that are transferred to the function as \texttt{VAR\_INPUT} are valid. Otherwise the function cannot do anything with them and returns \texttt{INVALIDPARAM} as the return code.

\begin{verbatim}
IF ipSrcImage = 0 THEN
  F\_CountAndDrawObjects := Tc2\_System.E\_HRESULTAdrErr.INVALIDPARAM;
  RETURN;
END_IF
\end{verbatim}

If all input checks were successful, the reference counter of all interface pointers transferred as \texttt{VAR\_INPUT} must be incremented, because due to the transfer of the interface pointer a copy is created that is retained after the function block is executed and has to be mapped in the reference counter. The method \texttt{TcAddRef} can be executed here directly without a prior \texttt{<> 0} check, as this has already taken place during the input checks. As already described in the declaration part, incrementing the reference counter is not necessary for functions and methods, because the variable values, like in this case the pointer address, are not preserved after execution.

Main processing

The actual image processing sequence is located in the main part of the function. This is only an example here and will not be considered any further. The call of \texttt{F\_VN\_ConvertColorSpace} illustrates that interface pointers, which are transferred as \texttt{REFERENCE TO} for the purpose of the result return, can be written directly by TwinCAT Vision API functions.

\begin{verbatim}
hr := F\_VN\_ConvertColorSpace(ipSrcImage, ipDestImage, TCVN\_CST\_GRAY\_TO\_RGB, hr);
stParams.bFilterByArea := TRUE;
stParams.fMinArea := 10_000;
stParams.fMaxArea := 100_000;
hr := F\_VN\_DetectBlobs(ipSrcImage, ipContours, stParams, hr);
hr := F\_VN\_GetNumberOfElements(ipContours, nNumberOfObjects, hr);
hr := F\_VN\_DrawContours(ipContours, -1, ipDestImage, aColor, 5, hr);
\end{verbatim}

By means of this image processing sequence, objects are painted in the result image and their quantity returned as \texttt{ULINT}:
Following the actual image processing sequence, the interface pointers declared within the function are released again. This is the case because all variables declared in the function (i.e. the interface pointers too) are deleted after ending the function. If there are still data in the background that have not yet been released, this will lead to memory leaks.

FW_SafeRelease(ADR(ipContours));

Finally, the HRESULT is assigned to the function as the return value in order to report any errors to the outside. So that it is not falsified, the return values of the FW_SafeRelease functions should not be assigned to the HRESULT.

F_CountAndDrawObjects := hr;

8.2 Function block samples

This section contains samples for the function blocks [1281].

8.2.1 Camera Register Access

8.2.1.1 Reading/writing a register

All camera parameters that can be read or written via the Configuration Assistant in Config mode can also be read from the PLC via the function blocks FB_VN_ReadRegister_UDINT [1287] and FB_VN_ReadRegister_REAL [1285] or written via FB_VN_WriteRegister_UDINT [1295] and FB_VN_WriteRegister_REAL [1293]. The function block required depends on the data type of the parameter and can be read in the Property list on the Register Type. In this sample, the exposure time is written and read as an integer.

fbReadValue : FB_VN_ReadRegister_UDINT;
fbWriteValue : FB_VN_WriteRegister_UDINT;

Both function blocks require the address of the parameter and an indication of the byte sequence (endianness). The Write function block additionally requires the parameter value that is to be written to the camera. As the value displayed or entered under Value need not directly conform to how the camera processes it internally, e.g. in the case of enums or other byte sequences, this can be read under the Value register.

This information can be read from the Configuration Assistant:
Camera-specific parameters

The parameter names, properties and values shown here are vendor-specific or camera-specific and firmware-specific and may therefore differ.

Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bTriggerReadValue</td>
<td>BOOL</td>
</tr>
<tr>
<td>nReadValue</td>
<td>UDINT</td>
</tr>
<tr>
<td>nReturnCodeRead</td>
<td>UDINT</td>
</tr>
<tr>
<td>bTriggerWriteValue</td>
<td>BOOL</td>
</tr>
<tr>
<td>nWriteValue</td>
<td>UDINT</td>
</tr>
<tr>
<td>nReturnCodeWrite</td>
<td>UDINT</td>
</tr>
</tbody>
</table>

Code

```c
fbReadValue(
    nAddress := 16#20004BFC, // Check the Address of the Camera Parameter
    nEndian := 1,          // 0 = Big, 1 = Little
    bRead   := bTriggerReadValue,
    nTimeout := T#5S,
    nValue  => nReadValue);
```

```c
fbWriteValue(
    nAddress := 16#20004BFC, // Check the Address of the Camera Parameter
    nValue   := nWriteValue, // Check the right Input Format
    nEndian := 1,           // 0 = Big, 1 = Little
    bWrite  := bTriggerWriteValue,
    nTimeout := T#5S);
```

Conditions for the reading and writing of parameters

Parameters can generally only be read or written with an opened camera connection (ETcVnCameraState => TCVN_CS_OPENED). Please refer to the respective vendor documentation for the camera for the exact conditions.

Both function blocks must be initialized with the Image Provider of the camera:
8.2.2 File Access

8.2.2.1 Saving images from the PLC

In this sample, images from the PLC are saved to a local drive. The function block FB_VN_WriteImage is used for this purpose.

The save process takes place asynchronously over several cycles.

Duration of the save process

The duration of the save process depends on various factors and can therefore not be stated in general terms.

Variables

hr : HRESULT;
fbCamera : FB_VN_SimpleCameraControl;
ipImageIn : ITCvImage;
ipImageInDisp : ITCvDisplayableImage;

// Sample Specific Variables
fbWriteImage : FB_VN_WriteImage := (nTimeout := T#500MS);
sFilePath : STRING(255) := '';
bWriteImageTrigger : BOOL;
bWriteImageWaitResult : BOOL;
bWriteImageDone : BOOL;
nReturnCode : UDINT;

In this sample an image can be saved by setting the trigger variable bWriteImageTrigger to TRUE.

Code

The ipImage transferred is already completely accepted on the first call of fbWriteImage with a rising edge at bWrite and can therefore be released immediately afterwards, e.g. with F_VN_TransformIntoDisplayableImage. Thereafter, the fbWriteImage must continue to be cyclically called until the bBusy output is FALSE again. This can be the case with an error or even with the first call, e.g. if no file extension or an incorrect one is specified. An evaluation of whether writing was successful or whether an error occurred takes place in the IF block below. If successful, bWriteImageDone is set and remains unchanged until the next time bWriteImageTrigger is set.

IF SUCCEEDED(hr) AND ipImageIn <> 0 THEN

    IF NOT fbWriteImage.bBusy AND bWriteImageTrigger THEN
        bWriteImageTrigger := FALSE;
        bWriteImageDone := FALSE;
        bWriteImageWaitResult := TRUE;
    
    IF NOT (fbWriteImage.bBusy AND bWriteImageTrigger) THEN
        IF NOT fbWriteImage.bBusy THEN
            bWriteImageTrigger := FALSE;
            bWriteImageDone := FALSE;
            bWriteImageWaitResult := TRUE;
        
    // With setting sFilePath:= '' to an empty string the images are saved under the default...
8.2.2.2 Serial loading of images into the PLC

In this sample you load images serially into the PLC. For this you use:

- **FB_VN_ReadImage**

This sample applies analogously to the loading of containers with **FB_VN_ReadContainer**.

**Explanation**

A frequent use case is where reference images are saved for comparison with the live image in the controller. These reference images should be replaced during operation, e.g. when a new reference is introduced. Images may be reloaded any number of times and at any interval.

So as not to have to inform the PLC of the file name for every new reference image, define a naming convention. This could be, for example, `Image<Index>.png`, where `<Index>` is to be replaced by a sequential integer. Each new reference image is then given a file name with an index incremented by 1. The PLC thus knows what the name of the next reference image must be and can purposefully attempt to load it.

**NOTE**

**Risk of file corruption**

Never replace the content of an image file; instead, only create new images in the file system! Firstly, replaced images are not registered due to internal caching mechanisms of the Vision service. Secondly, simultaneous access to an image by several processes can lead to a file corruption. The cache settings can be adjusted alternatively.

**Application**

6 color images with the name `Image<Index>.png` and the empty folder `load` are attached to this sample. Define the absolute path of the `load` folder as `sBasePath`. After starting the sample PLC, move the images in order (starting with `Index=0`) to the `load` folder and observe in the ADS Image Watch that the image `ipImageRefDisp` changes accordingly.

**Program**

In the cyclic program you initially calculate the path of the file to be loaded, based on the index `nIndex`. Also, adjust the base path `sBasePath` in the variable declaration so that it matches the memory location of your images.

```plaintext
sPath := CONCAT(CONCAT(sBasePath, TO_STRING(nIndex)), '.png');
```
Using the calculated path, cyclically call the function block fbReadImage of the type FB_VN_ReadImage [1304].

```
fbReadImage(
  sFilePath := sPath,
  ipDestImage := ipImageIn,
  bRead := TRUE,
  nTimeout := T#1S
);
```

In addition to calling the function block, you must react when it is ready. On the one hand, you reset the function block by means of a call with `bRead:=FALSE`; on the other, you increment the index and forward the image to the desired interface pointer, provided the execution is successful. Querying whether an error has occurred is very important, as you may only increment the index if the current image has already been loaded successfully.

```
IF NOT fbReadImage.bBusy THEN
  IF NOT fbReadImage.bError THEN
    nIndex := nIndex + 1;
    IF ipImageIn <> 0 THEN
      FW_SafeRelease(ADR(ipImageRef));
      ipImageRef := ipImageIn;
      ipImageIn.TcAddRef();
      FW_SafeRelease(ADR(ipImageIn));
    END_IF
  ELSE
    nReturnCode := fbReadImage.nErrorId AND 16#FFF;
  END_IF
  fbReadImage(sFilePath:='', bRead:=FALSE);
END_IF
```

In order to save memory and computing time, the forwarding of the image takes place here via an orderly transfer of the interface and not via a copy.

Also see about this

- Service Configuration [63]

### 8.2.3 Image acquisition

#### 8.2.3.1 Triggering an image by file name

In this sample an image is purposefully triggered on the basis of the file name in the function block FB_VN_FileSourceControl [1316].

**Application**

The sample displays alternately two images with the file names `Image1.bmp` and `Image2.bmp` from the file source in ADS Image Watch [125]. The sample includes the images with corresponding file names. You can replace the sample images with your own images or add them additionally. To do this, change or add to the file names in `sFileName` and adjust the number of elements if necessary. Note that all images to be displayed must be added in the File Source Control [120]. By setting `bLoop` to `false` you stop the automatic switching of images and then you can select the image to be triggered by setting `nImageIndex`.

**Program**

```
hr : HRESULT;
sFileName : ARRAY [0..1] OF STRING := ['Image1.bmp', 'Image2.bmp'];
fbFileSource : FB_VN_FileSourceControl;
eState : ETCvNCameraState;
ipImageIn : ITCvNImage;
ipImageInDisp : ITCvNDisplayableImage;
bTrigger : BOOL := TRUE;
bLoop : BOOL := TRUE;
nImageIndex : UINT := 0;
```
**fbFileSource**

Function block for controlling the File Source of the type **FB_VN_FileSourceControl**.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eState</td>
<td>State of the File Source</td>
</tr>
<tr>
<td>ipImageIn</td>
<td>Input image</td>
</tr>
<tr>
<td>ipImageInDisp</td>
<td>Displayable input image</td>
</tr>
<tr>
<td>bTrigger</td>
<td>Trigger flag</td>
</tr>
<tr>
<td>sFileName</td>
<td>File name – change this variable in order to load a different image. However, this image must be added in the File Source Control.</td>
</tr>
<tr>
<td>hr</td>
<td>Status variable of the type <strong>HRESULT</strong>.</td>
</tr>
</tbody>
</table>

In general, the usual state machine is used to operate the file source:

```plaintext
eState := fbFileSource.GetState();
CASE eState OF
<...>
END_CASE
```

Resetting in case of errors and the starting of the image acquisition take place just as normal in the corresponding states:

```
TCVN_CS_INITIAL, TCVN_CS_INITIALIZING, TCVN_CS_INITIALIZED, TCVN_CS_OPENING, TCVN_CS_OPENED,
TCVN_CS_STARTACQUISITION:
    fbFileSource.StartAcquisition();
TCVN_CS_ERROR:
    fbFileSource.Reset();
```

However, so that the file source doesn't load images immediately when the image acquisition is started, the **Trigger Mode** must be activated in the File Source Control:

```
TCVN_CS_ACQUIRING:
    IF bTrigger THEN
        hr := fbFileSource.TriggerImageByName(sFileName[nImageIndex]);
        IF SUCCEEDED(hr) THEN
            bTrigger := FALSE;
```

In the **ACQUIRING** state, the image with the corresponding file name will now be alternately triggered and then received. The method **TriggerImageByName** is explicitly used here to trigger the image. This method exists only in the function block **FB_VN_FileSourceControl** and not in **FB_VN_SimpleCameraControl**.
As the File Source temporarily enters the TRIGGERING state when triggering the image, triggering must continue in this state. This continues until the File Source has fully loaded the image and thus changed its state back to ACQUIRING. The file name of the image does not need to be transferred again for further triggering; it is sufficient to call the method `TriggerImage()`.

To run this sample, download the sample project with the created camera via Download. Open this project and open the Calibration Assistant of the camera instance.

### 8.3 Samples of assistants

#### 8.3.1 Calibration Assistant

A geometric camera calibration is required to convert image points to world points, during which the required parameters are determined. This sample shows the geometric calibration of a camera using an asymmetric circle pattern and the Calibration Assistant. The steps can also be performed analogously for the other calibration patterns (see Camera Calibration). Only step 2 has to be adapted accordingly.

To run this sample, download the sample project with the created camera via Download. Open this project and open the Calibration Assistant of the camera instance.
Step 1 - Image acquisition

First, images of the calibration pattern taken by the camera to be calibrated must be loaded into the Assistant. This can be done either by direct image acquisition with the camera or by loading existing images. For this sample, the _ImagesCalibration folder contains four suitable images of an asymmetric circle pattern. Load these images into the Calibration Assistant via **Load Images**.

Step 2 - Define asymmetric circle pattern

In this sample an asymmetric circle pattern is used as calibration pattern. In order to use one of the other pattern types, observe the respective definition:

- Chessboard pattern [111]
- Symmetrical circles pattern [112]
- Asymmetrical circles pattern [113]
- Individual circles pattern [114]

The asymmetric circle pattern must be selected in the Calibration Assistant for the loaded sample images:
Due to the asymmetry, either the first (x1) or the second (x2) row of the circle pattern is indented. In this case the first (x1) row is indented, so **Start Indent** should be selected.

![Image of circle pattern with x1 and x2 markings]

The width of the **Width** pattern corresponds to the number of points/circles in a row, while the **Height** corresponds to the number of rows. **Distance X** is used to specify the shortest distance between two circles in X direction. In the asymmetrical pattern, this is the distance to the next circle in the row below / above. Note the marking in the image. The distance between two rows is specified in **Distance Y**.

![Image of grid pattern with Distance X and Distance Y markings]

The Assistant expects black circles on white background. Since this example uses a pattern of white circles on a black background, the color must be inverted by selecting **Color inverted**.
Step 3 - Calculate intrinsic parameters

Once the calibration pattern has been fully defined, the intrinsic parameters can be calculated using the Calibrate Intrinsics button. The Reproj. Error should then be between 0 and 1, and the camera matrix and the distortion coefficients should be determined.

<table>
<thead>
<tr>
<th>Camera Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>221.834  0.000  400.000</td>
</tr>
<tr>
<td>0.000  221.834  300.000</td>
</tr>
<tr>
<td>0.000  0.000  1.000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Distortion Coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.07765, -1.61999, 0.00104, -0.00131, 36.24943, 0.00000, 0.00000, 0.00000</td>
</tr>
</tbody>
</table>

If this is not the case, the tooltip for the Reproj. Error provides information about the possible error. In addition, an image series is created under C:\TwinCAT\3.1\Components\Vision\CalibrationAssistantOutput that provides further information on the recognition of the pattern. Write access for the directory is required.

Step 4 - Calculate extrinsic parameters

After the intrinsic parameters the extrinsic parameters can be calculated. Only one image is used for this purpose, namely the image where the pattern is located in the plane on which the components are later measured, for example. This is the first of the sample images. Therefore, click on the first loaded image so that it appears in the image preview.

The zero point of the Extrinsic Origin pattern can also be specified. By default, this is in the center of the pattern.

Extrinsic Origin

The Calibrate Extrinsics button then calculates the rotation matrix and the translation vector. The results can be found under Results.
Step 5 - Write results to the image provider of the camera

If the calibration was successful, the results can be written to the Image Provider of the camera via Write Results. This step is necessary so that the camera function block FB_VN_GevCameraControl in the PLC can access the calibration results.

Results

![Image Provider Calibration Results]

In the Image Provider, the calibration results can be found under Parameters (Init) if Show Hidden Parameters is selected.

![Parameters Table]

After this step, make sure to activate the configuration by clicking on . Otherwise the calibration results are not available in the PLC.

Step 6 - Load calibration results into the PLC

The function block FB_VN_GevCameraControl can be used to load the calibration parameters into the PLC:

Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CameraMatrix</td>
<td>[(2316.27236754644, 0.0, 0.645.5)[0.0, 0.0, 2316.27236754644, 481.5][0.0, 0.0, 1.0]]</td>
</tr>
<tr>
<td>DistortionCoefficients</td>
<td>[0.153856158304233, -0.460737666748436, -0.005064833270091, -0.0002620...</td>
</tr>
<tr>
<td>RotationMatrix</td>
<td>[(0.999983196160105, 0.00262868175682932, -0.00051668286060862)[-0.002...</td>
</tr>
<tr>
<td>TranslationVector</td>
<td>[-2.15127341787928, -12.92785277996, 258.931580200648]</td>
</tr>
<tr>
<td>CalibPatternRef</td>
<td>[-29.25, -16.25, 0.0, -22.75, -16.25, 0.0, -16.25, -16.25, 0.0, -9.75, -16.25, 0.0, -...</td>
</tr>
<tr>
<td>ImageQueueSize</td>
<td>1</td>
</tr>
</tbody>
</table>
Step 7 - Transform pixels into world coordinates

Then use the function `F_VN_TransformCoordinatesImageToWorld_Points` to determine the coordinates of a point in the image in the real world in relation to the zero point of the calibration pattern, for example:

**Variables**

- `aPointImage` : `TcVnPoint2_LREAL`
- `aPointWorld` : `TcVnPoint3_LREAL`

**Code**

```cpp
hr := F_VN_TransformCoordinatesImageToWorld_Point(
    aSrcPoint := aPointImage,
    aDestPoint := aPointWorld,
    aCameraMatrix := aCameraMatrix,
    aDistortionCoefficients := aDistortionCoefficients,
    aRotationMatrix := aRotationMatrix,
    aTranslationVector := aTranslationVector,
    fZ := 0,
    hrPrev := hr);
```

Step 8 - Transform world points into image coordinates.

This transformation can be reversed with the function `F_VN_TransformCoordinatesWorldToImage_Points`, so that points from the world coordinate system are transformed in such a way that they can be displayed at the correct position in the image. This is useful for displaying auxiliary lines for better orientation in the image. The following code shows the world coordinate system in the image.

**Variables**

- `i` : `INT`
- `aLine` : `TcVnVector4_DINT`
- `aCoordinatesWorld` : `ARRAY[0..4] OF TcVnPoint3_LREAL := [ [0,0,0], [50,0,0], [-50, 0, 0], [0, 50, 0], [0, -50, 0] ];`
- `aCoordinatesImage` : `ARRAY[0..4] OF TcVnPoint2_REAL`

**Code**

```cpp
FOR i:= 0 TO 4 DO
    hr := F_VN_TransformCoordinatesWorldToImage_Point(
        aSrcPoint := aCoordinatesWorld[i],
        aDestPoint := aCoordinatesImage[i],
        aCameraMatrix := aCameraMatrix,
        aDistortionCoefficients := aDistortionCoefficients,
        aRotationMatrix := aRotationMatrix,
        aTranslationVector := aTranslationVector,
        hrPrev := hr);
    IF i > 0 THEN
        aLine[0] := REAL_TO_DINT(aCoordinatesImage[0][0]);
        aLine[1] := REAL_TO_DINT(aCoordinatesImage[0][1]);
        aLine[2] := REAL_TO_DINT(aCoordinatesImage[1][0]);
        aLine[3] := REAL_TO_DINT(aCoordinatesImage[1][1]);
        hr := F_VN_DrawLine_TcVnVector4_DINT(aLine , ipImageRes, aColorGreen, 3, hr);
    END_IF
END_FOR
```

Step 9 – View the results

When the program is executed, the result image can be viewed in `ADS Image Watch`. One can see the drawn coordinate axes and the determined world coordinates of the individual points on the image.
The result will be different if a different image or coordinate origin is used for the extrinsic calibration in step 4. The coordinate axes are shifted and the world coordinates of the pixels are also adjusted.
8.4 Camera configuration samples

Since the parameters of each GigE Vision camera may differ, it is not possible to provide a complete description. The following section provides an overview of frequently used camera parameters and how to configure them. Instructions for the configuration of GenAPI parameters can be found in the chapter on the Configuration.

8.4.1 Region of Interest (ROI)

If the camera sensor records a larger field of view than required, it is advisable to place a Region of Interest (ROI) on the actually required field of view:

- The sensor can be read out faster, depending on the camera sensor and the selected range.
- The amount of data to be transferred is reduced, which also reduces the data transfer time.
- At the same time, the processing time of all image processing algorithms, which take place over the entire image, is also reduced.
- In addition, image noise can be reduced, making image processing more robust and easier.

Checking the features of the camera

Please refer to the manual for the camera to see whether the camera supports corresponding features. The same applies to the feature names that are actually used. The procedure using the feature names according to the GenICam Standard Features Naming Convention (Version 2.4) is described below.
The features for setting a ROI can be found under `ImageFormatControl` or the corresponding category on your camera:

- `WidthMax` and `HeightMax` specify the maximum width and height of the image in pixels. These values may vary depending on other features (e.g. binning).
- `Width` and `Height` specify the width and height of the ROI, i.e. the field of view to be viewed, and must be set accordingly.
- `OffsetX` and `OffsetY` are used to position the ROI or the field of view. The starting point is the upper left corner of the image.

When setting the features, note that:

- `OffsetX + Width <= WidthMax`
- `OffsetY + Height <= HeightMax`

### 8.4.2 Binning

Binning is the summation or averaging of several pixel values into one.

It can be used to:

- increase the light sensitivity,
- improve the contrast,
- reduce the image resolution with the same field of view, enabling faster frame rates and processing times.

It should not be used if the required accuracies can no longer be achieved when the image resolution is reduced.

When summing up or averaging pixels, a distinction is made between horizontal and vertical binning.
Checking the features of the camera

Please refer to the manual for the camera to see whether the camera supports corresponding features. The same applies to the feature names that are actually used. The procedure using the feature names according to the GenICam Standard Features Naming Convention (Version 2.4) is described below.

The features for setting up binning can be found under `ImageFormatControl` or the corresponding category on your camera:

- `BinningSelector` indicates whether the binning applies to the entire sensor or an ROI
- `BinningHorizontalMode` and `BinningVerticalMode` indicate whether the pixels should be added up or averaged.

`BinningHorizontal` and `BinningVertical` indicate how many pixels should be combined in horizontal and vertical direction. A value of 1 indicates that there is no binning in the corresponding direction.

Binning with a monochrome camera

Four possible configurations for a monochrome camera are shown as examples (the coordinates of the pixels are shown):
Binning with a color camera

Binning support is rarer in color cameras than in monochrome cameras. It can be implemented in different ways. For details please refer to the camera user manual.
8.4.3 Image acquisition and trigger

The acquisition and trigger features are interrelated and are therefore be considered together below. The term **Acquisition** basically describes a state of cameras in which images can be taken, see [Image Acquisition](#). Whether and when images are actually taken and sent is determined by the configuration of the **Trigger** parameter.

The various configuration options and relationships between acquisition and trigger settings are described in more detail in the following chapters, which are arranged according to the **AcquisitionMode**.

By far the most commonly used image capture options are:

- **Continuous Image Acquisition** - the camera streams, it captures images at regular intervals without any other triggers.
- **Single image trigger with continuous image acquisition** - the camera takes exactly one image after a trigger signal (hardware or software). The camera remains in the state **ACQUIRING**, therefore it is not necessary to write the **AcquisitionStart** command repeatedly before each recording.

### Checking the features of the camera

Please refer to the manual for the camera to see whether the camera supports corresponding features. The same applies to the feature names that are actually used. The procedure using the feature names according to the GenICam Standard Features Naming Convention (Version 2.4) is described below.

#### Acquisition Mode

The **AcquisitionMode** feature mainly indicates how many images are taken during an acquisition.

- **Continuous**
  - Image acquisition begins with the **AcquisitionStart** command and is only terminated by the **AcquisitionStop** command. In between, images are continuously captured depending on the trigger settings.
  - **Continuous image acquisition**

- **MultiFrame**
  - Images are captured until the number of images specified in **AcquisitionFrameCount** is reached. A **AcquisitionStart** command is required for each image acquisition sequence.
  - **Multi-image acquisition**

- **SingleFrame**
  - A single image is captured. A **AcquisitionStart** command is required for each image acquisition.
  - **Single image acquisition**

#### Trigger Source

The **TriggerSource** feature specifies which internal or physical signal is to be interpreted as a trigger. The list of possible sources is long and varies depending on the camera. Some samples are provided below:

- **Software**
  - A software command sets the trigger. The PLC function blocks **FB_VN_GevCameraControl** and **FB_VN_SimpleCameraControl** provide the method **TriggerImage** for sending the command.

- **Line0, Line1, ...**
  - A physical camera input is the trigger source. Pay attention to the necessary edge steepness with the signal source connected to it. This may vary depending on the camera.
• Encoder0, Encoder1, ...
Line scan cameras are often connected to an encoder to trigger image acquisition. If no physical encoder is available, but the corresponding axis values are available in TwinCAT, the encoder signal can be output via an EL2521 or EL2522.

Trigger Activation

• RisingEdge
The trigger is activated with a rising edge of the TriggerSource.

\[
\text{Signal} \quad \uparrow \quad \uparrow
\]

\[
\text{RisingEdge} \quad \uparrow \quad \uparrow
\]

• FallingEdge
The trigger is activated with a falling edge of the TriggerSource.

\[
\text{Signal} \quad \downarrow \quad \downarrow
\]

\[
\text{FallingEdge} \quad \downarrow \quad \downarrow
\]

• AnyEdge
The trigger is activated with a rising or falling edge of the TriggerSource.

\[
\text{Signal} \quad \uparrow \quad \downarrow \quad \uparrow \\
\text{AnyEdge} \quad \uparrow \quad \downarrow \quad \uparrow
\]

• LevelHigh
The trigger is activated as long as a signal is present at the TriggerSource.

\[
\text{Signal} \quad \uparrow \quad \uparrow \\
\text{LevelHigh} \quad \uparrow \quad \uparrow
\]

• LevelLow
The trigger is activated as long as no signal is present at the TriggerSource.

\[
\text{Signal} \quad \uparrow \quad \uparrow \\
\text{LevelLow} \quad \uparrow \quad \uparrow
\]

8.4.3.1 Continuous image acquisition

The settings for image acquisition and triggering are interlocked by default. The following explanatory section assumes that the AcquisitionMode is set to Continuous.

The other cases are described under:

• Multi-image acquisition [1462]
• Single image acquisition [1464]
Checking the features of the camera

Please refer to the manual for the camera to see whether the camera supports corresponding features. The same applies to the feature names that are actually used. The procedure using the feature names according to the GenICam Standard Features Naming Convention (Version 2.4) is described below.

Continuous image acquisition

The following configuration is required if the camera is to stream, i.e. take images at even intervals without further triggers:

- AcquisitionMode = Continuous
- TriggerSelector = AcquisitionStart
- TriggerMode = Off

The AcquisitionStart command then starts the streaming and the AcquisitionStop command ends the streaming. Recording of the current image is completed, and the image is transferred.

Continuous image acquisition after trigger start signal

If you require the camera to continuously stream images after a trigger signal at even intervals, the following configuration is required:

- AcquisitionMode = Continuous
- TriggerSelector = AcquisitionStart
- TriggerMode = On
- TriggerSource = Lin1 (exemplarisch)

After the AcquisitionStart command, the camera is ready to capture and transfer images. However, the recording only begins after a trigger signal, in the sample after a hardware trigger signal on Line 1. Streaming is ended by the AcquisitionStop command; however, the recording of the current image is still completed and transferred.

Single image trigger for continuous image acquisition

If the camera is to capture just one image after a trigger signal, the following configuration is required:

- AcquisitionMode = Continuous
- TriggerSelector = FrameStart
- TriggerMode = On
- TriggerSource = Lin1 (exemplarisch)
As in the other cases, the camera is ready to take images after the AcquisitionStart command. Each image is recorded after a trigger signal from the selected source. Details of the trigger signal are specified in TriggerActivation.

Multi-image trigger for continuous image acquisition

If the camera is to record a certain number of images after a trigger signal, the following configuration is required. The number is set to 2 as an example:

• AcquisitionMode = Continuous
• AcquisitionBurstFrameCount = 2 (exemplarisch)
• TriggerSelector = FrameBurstStart
• TriggerMode = On
• TriggerSource = Line1 (exemplarisch)

As in the other cases, the camera is ready to take images after the AcquisitionStart command. The defined number of images is recorded after a trigger signal of the selected source. Details of the trigger signal are specified in TriggerActivation. If the AcquisitionStop command is issued during image acquisition/transfer, the image that has already been started is completed, but any missing images are no longer created.

8.4.3.2 Multi-image acquisition

The settings for image acquisition and triggering are interlocked by default. The following explanatory section assumes that the AcquisitionMode is set to MultiFrame.

The other cases are described under:

• Continuous image acquisition [1460]
• Single image acquisition [1464]

Checking the features of the camera

Please refer to the manual for the camera to see whether the camera supports corresponding features. The same applies to the feature names that are actually used. The procedure using the feature names according to the GenICam Standard Features Naming Convention (Version 2.4) is described below.

Multi-image acquisition

If the camera is to stream a certain number of images without using a trigger, the following configuration is required:

• AcquisitionMode = MultiFrame
• AcquisitionFrameCount = 2 (example)
• TriggerSelector = AcquisitionStart
• TriggerMode = Off

Streaming starts when the AcquisitionStart command is issued. It stops it when the specified number of frames (AcquisitionFrameCount) is reached. Optionally, the AcquisitionStop command can then be issued and the lock can be canceled. This makes it possible to change camera features before the next AcquisitionStart command. If the AcquisitionStop command is issued during image acquisition/transfer, the image that has already been started is completed, but any missing images are no longer created.

Multi-image acquisition after trigger start signal

If the camera is to stream a certain number of images after a trigger signal, the following configuration is required:

• AcquisitionMode = MultiFrame
• AcquisitionFrameCount = 2 (example)
• TriggerSelector = AcquisitionStart
• TriggerMode = On
• TriggerSource = Line1 (example)

The AcquisitionStart command activates streaming. However, the recording of the specified images only begins after a trigger signal, in the sample after a hardware trigger signal on Line 1. As with the multi-image acquisition (without trigger), the AcquisitionStop command and the cancelation of the lock can optionally take place after reaching the specified number of images.

Multi-image acquisition with single-image trigger

Configuration:

• AcquisitionMode = MultiFrame
• AcquisitionFrameCount = 2 (example)
• TriggerSelector = FrameStart
• TriggerMode = On
• TriggerSource = Line1 (example)
Multi-image acquisition with multi-image trigger

Configuration:
- AcquisitionMode = MultiFrame
- AcquisitionFrameCount = 3 (example)
- AcquisitionFrameBurstCount = 2 (example)
- TriggerSelector = FrameBurstStart
- TriggerMode = On
- TriggerSource = Line1 (example)

8.4.3.3 Single image acquisition

The settings for image acquisition and triggering are interlocked by default. The following explanatory section assumes that the AcquisitionMode is set to SingleFrame.

The other cases are described under:
- Continuous image acquisition [1460]
- Multi-image acquisition [1462]

Checking the features of the camera

Please refer to the manual for the camera to see whether the camera supports corresponding features. The same applies to the feature names that are actually used. The procedure using the feature names according to the GenICam Standard Features Naming Convention (Version 2.4) is described below.

Single image acquisition

If just one image is to be recorded after each AcquisitionStart command, the following configuration is required:
- AcquisitionMode = SingleFrame
- TriggerSelector = AcquisitionStart
- TriggerMode = Off

After the AcquisitionStart command, just one image is captured. Between two AcquisitionStart commands an AcquisitionStop command can be issued, and the lock can be canceled. Unlocking facilitates changing camera features between AcquisitionStart commands. No image is captured if an AcquisitionStop command is issued too soon after the AcquisitionStart command, when image acquisition has not yet started.
Single image acquisition with image acquisition trigger or single image acquisition with single image trigger

The following configurations can be used if just one image is to be recorded after each `AcquisitionStart` command in situations where a trigger signal is required:

**Configuration – single image acquisition with image acquisition trigger**
- AcquisitionMode = SingleFrame
- TriggerSelector = AcquisitionStart
- TriggerMode = On
- TriggerSource = Line1 (example)

**Configuration – single image acquisition with single image trigger**
- AcquisitionMode = SingleFrame
- TriggerSelector = FrameStart
- TriggerMode = On
- TriggerSource = Line1 (example)
Appendix

In this chapter you will find additional information such as FAQs, an overview of the paths used by TwinCAT Vision and a chapter on troubleshooting for the rectification of known problems.

9.1 FAQ

What is TwinCAT Vision?

TwinCAT Vision is an industrial image processing software. You can capture images with industrial cameras and calculate information about the recorded scene from them.

Frequent use cases are:
- Presence check
- Object recognition
- Optical measurement
- Code Reading

Can I test TwinCAT Vision free of charge?

Yes, you can generate 7-day trial licenses for the runtime components. The engineering is free of charge in any case.

Do I need TwinCAT in order to use TwinCAT Vision?

Yes, TwinCAT Vision is fully integrated in TwinCAT. A compatible TwinCAT version must be installed before TwinCAT Vision can be installed.

What cameras can I use with TwinCAT Vision?

TwinCAT Vision supports cameras with the GigE Vision interface. GigE Vision cameras are connected to the IPC via an Ethernet cable.

Can I use several cameras at the same time?

Yes, with the TwinCAT GigE Vision Connector you can operate several cameras, depending on the license.

Can I connect several cameras via a single switch?

Yes. Pay attention to the maximum available bandwidth (e.g. 125 MB/s in case of 1 GigE, effectively nearer 110 MB/s).

Are there samples for TwinCAT Vision?

Yes, in the chapter Samples [1362].

Are there benchmarks for the TwinCAT Vision functions?

No, because the execution time of Vision algorithms very much depends on the function parameters and the image content. It is therefore helpful to have a basic understanding of how algorithms work instead of concerning yourself with benchmark tables.

Can other image processing software also be used in TwinCAT as an alternative to TwinCAT Vision?

Images can be recorded via GigE Vision with the TC3 GigE Vision Connector. The image processing can then take place in various ways:
- Use of the TwinCAT Vision API via the PLC library Tc3_Vision
• Use of the self-written TwinCAT C++ algorithms
• In the future: Use of the TwinCAT Vision API via a C++ interface
• In the future: Use of Matlab/Simulink

**What is the difference between image processing software such as TwinCAT Vision and smart cameras?**

Image processing software such as TwinCAT Vision:
• Image data received from a camera
• Process this on a computer
• Send processed information to the controller
• Advantage of TwinCAT Vision: Controller and image processing are the same software and run on the same system. Therefore, additional interfaces and protocols are dispensed with.

Smart cameras:
• Have an internal processor
• Carry out image processing operations directly on the camera
• Supply processed information such as good/bad information or the position of a workpiece via additional interfaces

**Can TwinCAT Vision be executed on a GPU?**

No, TwinCAT Vision runs exclusively on a CPU in the TwinCAT real-time environment. Alternatively, TwinCAT Vision offers the option of an automatically parallelized execution by using the multi-core functionality of TwinCAT (in interaction with job tasks [61]).

### 9.2 Important paths

**Paths for Vision data**

<table>
<thead>
<tr>
<th>NOTE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Paths that are relevant for data</strong></td>
</tr>
<tr>
<td>Make sure that the following standard paths are not affected by Write filters or similar programs. Failure to do so may result in data loss. If the corresponding path settings were adjusted manually, the same applies to the set paths.</td>
</tr>
</tbody>
</table>
### Appendices

**TF7100 - TF7300**

**Version:** 1.3

---

#### Use | Path
--- | ---
Storage location for File Source images | `C:\Users\Public\TcVision\FileSources\<UniqueId>`
(Automatically defined by the File Source. `<UniqueId>` is different for every project.)

Storage location for camera streams | `C:\Users\Public\TcVision\CameraStreams`
(May differ depending on the setting in Record/Playback [p. 97].)

Storage location for images that are read or written by PLC functions. | `C:\Users\Public\TcVision\Images`
(May differ depending on the setting in Service Configuration [p. 63].)

Storage location for containers that are read or written by PLC functions. | `C:\Users\Public\TcVision\Containers`
(May differ depending on the setting in Service Configuration [p. 63].)

Storage location for ML models that are read or written by PLC functions. | `C:\Users\Public\TcVision\MLModels`
(May differ depending on the setting in Service Configuration [p. 63].)

Storage location for individual calibration patterns | `%TWINCAT3DIR%\3.1\CustomConfig\Vision\CalibrationPattern`

---

#### Paths for log files

| Log file | Path |
--- | --- |
TwinCAT Vision installation log (all other TwinCAT installation logs are also located in this folder) | `C:\Users\<User>\TwinCAT-Vision.log`
This log file is located in the BeckhoffSetupLogFiles.zip folder.

TwinCAT Vision Service Log | `C:\ProgramData\Beckhoff\TcVnService\TcVnService.log`

General TwinCAT Vision log | `%TWINCAT3DIR%\3.1\Components\Vision.log`
(May differ depending on settings in Logging [p. 64].)

Result images of the Calibration Assistant [p. 103] for analyzing the recognition of feature points on the calibration pattern | `%TWINCAT3DIR%\3.1\Components\Vision\_CalibrationAssistantOutput`

---

### 9.3 Troubleshooting

Known error situations and corresponding solutions are described below. If you can attribute a problem to a specific function or function block, first read the corresponding entry in the API reference [p. 131].

#### 9.3.1 TwinCAT system start

Solutions to situations in which the start of the TwinCAT system is prevented are explained below.

#### 9.3.1.1 TwinCAT system does not start

If the TwinCAT system cannot be started, the following message frequently appears. On the basis of the error code with the red border, instructions for rectifying the problem can be given. Further below you will find information on the usual error codes with TwinCAT Vision applications.
Invalid Object ID

<table>
<thead>
<tr>
<th>Error message</th>
<th>Solution 1</th>
</tr>
</thead>
</table>
| No task linked | 1. Check whether the corresponding tasks have been created for each Vision device used and add missing tasks.  
  ◦ Check whether a task for image acquisition has been created for each File Source Control (default cycle time 10 ms).  
  ◦ Check whether each GigE Vision camera is associated with a task for Image Acquisition (default cycle time 1 ms) and, if there is an Ads Communicator, this is also associated with a task (default cycle time 10 ms).  
  2. The tasks must be assigned to the corresponding TcCom modules of the Vision devices. Check the assignment and correct it if necessary.  
  ◦ Check whether the Image Provider Task of the File Source is linked to the Image Provider of the File Source Control.  
  ◦ Check whether the Image Acquisition Task of the camera is linked to the Image Acquisition and Image Acquisition Simulation of the camera.  
  ◦ Link the Ads Communicator Task with the Ads Communicator of the camera.  
  ◦ Do not assign a task to the Image Provider of the camera. |

<table>
<thead>
<tr>
<th>Solution 2</th>
<th>Network adapter invalid</th>
</tr>
</thead>
</table>
|            | The network adapter was not adapted, e.g. following a system change. Adapt the network adapter to the current target system.  
In the respective network adapter, open the Adapter tab and select a new compatible adapter with Search… |

<table>
<thead>
<tr>
<th>Solution 3</th>
<th>Network adapter deleted</th>
</tr>
</thead>
</table>
|            | The network adapter was deleted, for example, on the assumption that it is not required for the simulation of the camera. However, each camera object always requires a network adapter; if necessary it can be deactivated but not deleted.  
Create a new network adapter with an IpStack and link the IpStack on the Interface Pointer tab of the acquisition object of the camera. |

Further information on the linking of Vision TcCOM objects can be found in the corresponding chapters for the camera [117] and the File Source [124].
## General ADS Error

<table>
<thead>
<tr>
<th>Error message</th>
<th>ADS ERROR: General ADS Error</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Solution 1</strong></td>
<td><strong>More cameras than configured</strong></td>
</tr>
<tr>
<td></td>
<td>The default setting of the network adapter allows up to 2 cameras to be operated simultaneously. If you wish to operate more cameras on the same adapter, increase the Init parameters <code>ipMaxReceiver</code> and <code>UdpMaxReceiver</code> to twice the number of connected cameras (see <code>Configuration of the network adapter</code> [54]).</td>
</tr>
</tbody>
</table>
## 9.3.1.2 Vision Service doesn't start

<table>
<thead>
<tr>
<th>Situation</th>
<th>The TwinCAT Vision Service doesn't start. This can be seen by the connection status &quot;Offline&quot; on the <strong>Service Configuration</strong> tab in the VISION node.</th>
</tr>
</thead>
</table>
| **Vision Service Status** | **Service on Target Machine**: `<Loca>` *(172.17.54.233 1.1)*  
**Connection**: **Offline** |  
**Version**: |

<table>
<thead>
<tr>
<th>Solution</th>
<th>If the TwinCAT Vision Service only fails to start once or fails, it can be reactivated with a TwinCAT restart (click <img src="image" alt="click" />). However, if it generally fails to start (after the system startup), the service must be re-registered as a TwinCAT service. To do this, follow these steps:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td><strong>Switch TwinCAT to the Config mode</strong> <img src="image" alt="config_mode" /></td>
</tr>
<tr>
<td>2.</td>
<td><strong>Open a prompt as an administrator.</strong> <img src="image" alt="command_prompt" /></td>
</tr>
<tr>
<td>3.</td>
<td><strong>Change to the corresponding folder with</strong> &quot;cd C:\TwinCAT\3.1\Components\Vision\TcVision&quot;.</td>
</tr>
<tr>
<td>4.</td>
<td><strong>Execute the command</strong> &quot;TcVnService.exe /tc3serverreg&quot; to start the Vision Service.</td>
</tr>
<tr>
<td>5.</td>
<td><strong>Confirm the subsequent dialog by clicking <strong>Yes</strong>.</strong> <img src="image" alt="yes_no_dialog" /></td>
</tr>
</tbody>
</table>

![Yes](image) The Vision Service is started.  
**Note**: **TwinCAT must be in Config mode so that the Vision service can be started. Therefore, the machine must be restarted if the Vision service is to be restarted.** If the Vision service is to be manually stopped, you can use the command "TcVnService.exe /unregtc3server".
9.3.1.3  Binary or repository could not be found/loaded

**Situation**
One of the two messages appears when the configuration is activated. Depending on the cause and the installed version, other file names or version numbers may appear in the message.

The reason for the message is that the version used in the project does not match the version installed on the system.

---

**Solution 1**  
**Checking the Vision Library**
Check the version of the Tc3_Vision library included in the project. If the version does not match the one in the folder `C:\TwinCAT\3.1\Repository\Beckhoff Automation GmbH\Tc3_Vision` of the engineering system, either update the library in the project or install another setup version.

**Solution 2**  
**Checking the TMC version of the camera or FileSorce**
Check under `SYSTEM > TcCOM Objects > Project Objects` the TMC version of the listed objects used in the project. If the version is highlighted in yellow, the specified version does not exist on the system or does not match.

Now either update the corresponding TMC versions (select the version via pull-down of the version, then right click on one of the object IDs in the first column and select `Reload TMI/TMC Description(s) with changed version`) to the available version or install another setup version.

**Solution 3**  
**The vision component is not available**
The corresponding repository file is missing on the current system because the vision component was not selected during installation. Basically, the versions of the engineering and target systems must match. Therefore, if the systems are different, check both installations. On pure runtime systems XAR, only the Vision Service can be installed since all other files are transferred by the engineering system XAE. Install the missing vision component or run the setup with the repair option.

---

9.3.2  PLC runtime environment

Solutions for situations in which errors occur when the PLC is running are explained below.
9.3.2.1 Vision device in the ERROR state

Camera function blocks switch between the ERROR and INITIAL state

<table>
<thead>
<tr>
<th>Situation</th>
<th>The instance of the function block FB_VN_Gev_CameraControl [1324] or FB_VN_SimpleCameraControl [1348] alternately returns TCVN_CS_ERROR and TCVN_CS_INITIAL on calling the method GetState. In the TCVN_CS_ERROR state, the Reset() method of the function block is executed successfully and the camera instance changes to the TCVN_CS_INITIAL state. If the camera remains in the TCVN_CS_ERROR state, refer to Device is permanently in ERROR state [1474].</th>
</tr>
</thead>
<tbody>
<tr>
<td>eState := fbCamera.GetState()</td>
<td>IF eState = TCVN_CS_ERROR THEN fbCamera.Reset(); END_IF</td>
</tr>
<tr>
<td>In the TCVN_CS_INITIAL state, either function block method OpenCamera() or StartAcquisition() is called. The call fails and the instance returns to state TCVN_CS_ERROR.</td>
<td>eState := fbCamera.GetState() IF eState = TCVN_CS_INITIAL THEN fbCamera.OpenCamera(); END_IF or eState := fbCamera.GetState() IF eState = TCVN_CS_INITIAL THEN fbCamera.StartAcquisition(); END_IF</td>
</tr>
</tbody>
</table>

Solution 1 Check the physical connection to the camera and the power supply of the camera.

Solution 2 Check the settings for the camera connection (IP address of the camera, network port).
Function block permanently in the ERROR state
Situation

The instance of the function block FB_VN_GevCameraControl, FB_VN_SimpleCameraControl or FB_VN_FileSourceControl permanently returns the state TCVN_CS_ERROR. Even when calling the function block method Reset, there is no change of state to TCVN_CS_INITIAL (if there is, see Device switches between ERROR and INITIAL state [1473]).

```
fbCamera.Reset();
```

Solution 1

The symbol instance must be linked to the Image Provider - each instance of an FB_VN_GevCameraControl, an FB_VN_SimpleCameraControl and an FB_VN_FileSourceControl must be linked to a camera or File Source Control created in the Vision node.

Proceed as follows:

1. Perform a successful rebuild of the PLC project.
2. Then double-click on the instance of the PLC project.
3. Under the Symbol Initialization tab, assign the Image Provider of the corresponding camera or the corresponding File Source Control to the instances of FB_VN_GevCameraControl.
4. Reactivate the configuration.

**Note** The FB_VN_GevCameraControl can only be linked to the Image Provider of a camera.

**Note** The FB_VN_FileSourceControl can only be linked to the Image Provider of a File Source Control.

**Note** The FB_VN_SimpleCameraControl can be linked to the Image Provider of a camera or to the Image Provider of a File Source Control.

Solution 2

The instance of FB_VN_GevCameraControl or FB_VN_SimpleCameraControl or FB_VN_FileSourceControl is assigned the Image Provider of a camera or a File Source Control, but the device is permanently in error state.

Check the method calls of the instance of FB_VN_GevCameraControl or FB_VN_SimpleCameraControl or FB_VN_FileSourceControl. Many methods may only be called in a certain state: A detailed description of the states of Vision devices can be found in the section Function blocks for image acquisition [1314].
• In all states
  • GetState
  • GetCalibPatternRef (from FB_VN_GevCameraControl)
  • GetCameraMatrix (from FB_VN_GevCameraControl)
  • GetDistortionCoefficients (from FB_VN_GevCameraControl)
  • GetRotationMatrix (from FB_VN_GevCameraControl)
  • GetTranslationVector (from FB_VN_GevCameraControl)
  • SetCameraMatrix (from FB_VN_GevCameraControl)
  • SetDistortionCoefficients (from FB_VN_GevCameraControl)
  • SetRotationMatrix (from FB_VN_GevCameraControl)
  • SetTranslationVector (from FB_VN_GevCameraControl)
• In the INITIAL state and in the INITIALIZING state
  • InitializeCamera (from FB_VN_GevCameraControl)
• In INITIAL state and in OPENING state
  • OpenCamera (from FB_VN_GevCameraControl)
  • StartAcquisition
• In OPENED state and in STARTACQUISITION state
  • StartAcquisition
• In OPENED state and in CLOSING state
  • CloseCamera (from FB_VN_GevCameraControl)
• In ACQUIRING state
  • GetCurrentImage
  • GetCurrentImageUndistorted (from FB_VN_GevCameraControl)
  • GetCurrentImageWithGvspInfo (from FB_VN_GevCameraControl)
  • GetCurrentImageAndFileName (from FB_VN_FileSourceControl)
  • ClearImageQueue (from FB_VN_GevCameraControl)
  • GetLastImageFromQueue (from FB_VN_GevCameraControl)
  • GetOmittedImagesNum (from FB_VN_GevCameraControl)
• In ACQUIRING state and TRIGGERING state
  • TriggerImage
  • TriggerImageExp (from FB_VN_FileSourceControl)
  • TriggerImageByName (from FB_VN_FileSourceControl)
• In ACQUIRING state and in STOPACQUISITION state
  • StopAcquisition
• In ERROR state
  • Reset

9.3.2.2 Camera not found after restart

<table>
<thead>
<tr>
<th>Situation</th>
<th>Application is working entirely as usual. No connection can be established with the camera after restarting the machine.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solution 1</td>
<td>The IP address has changed. Use the Force-IP option in the Configuration Assistant [71] of the camera.</td>
</tr>
</tbody>
</table>
9.3.2.3 Error when starting a watchdog

<table>
<thead>
<tr>
<th>Situation</th>
<th>The function <em>F_VN_StartAbsWatchdog</em> ([\text{878}]) or <em>F_VN_StartRelWatchdog</em> ([\text{880}]) permanently returns the error code (16#71A) (NOINTERFACE).</th>
</tr>
</thead>
</table>
| | ```c
hrWD := F_VN_StartRelWatchdog(5000, 0);     
hr := F_VN_DetectElobo(ipImgIn, 16#FFFF0000A072D); 
hrWD := F_VN_StopWatchdog(hrWD, 16#811071A, tRest); 
``` |
| Solution | Activate the watchdog stack in the corresponding PLC task and then re-activate the configuration. |
9.3.2.4 Vision Error: Assertion failed

<table>
<thead>
<tr>
<th>Error message</th>
<th>During the execution of a Vision application, an error message similar to the following appears. At the same time, TwinCAT enters the EXCEPTION state.</th>
</tr>
</thead>
</table>

This message indicates that there is a conflict in the execution of a Vision function that could not be intercepted. This essentially refers to the correct format of the input data and should only occur if the function is not used in accordance with the API reference. The error specification behind "VisionError: Assertion failed" may vary depending on the use case.

Solution with F_VN_ConvertColorSpace [1083]

An image can be either single-channel or multi-channel. For certain function calls, an image with a defined number of channels is expected. For example with the function F_VN_ConvertColorSpace, where the enum ETcVnColorSpaceTransform is used to specify the format of the input image. If Engineering is logged in, the corresponding function call is marked in the PLC code. The function F_VN_GetPixelFormat, which returns the actual number of channels, can help to analyze why the actual number of channels differs from the expected number.

Typical causes:
- File Source Control
  - In File Source a different image format is set than expected or the image format is set to "Original Format from File" and the actual image format is different than expected. For example, images that look monochrome are not always stored as monochrome images (single-channel image).
- Camera
  - For many cameras, the pixel format can be set in the configuration. Depending on the selected format, the number of image channels also changes.
- Previous processing steps
  - A previous function call may have changed the image and thus also the number of channels.

9.3.2.5 Error: SSE Invalid Operation

<table>
<thead>
<tr>
<th>Error message</th>
<th>You get an error message called SSE Invalid Operation.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solution</td>
<td>This error can occur unnecessarily if the Floating point exceptions handling is activated on the executing task. Follow the instructions in the section CPU cores and tasks [58], by deactivating the option Floating point exceptions.</td>
</tr>
</tbody>
</table>
9.3.2.6 File Source

Error state

<table>
<thead>
<tr>
<th>Error pattern</th>
<th>Solution 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>The File Source constantly switches to the error state when the PLC is running.</td>
<td>Too little memory</td>
</tr>
<tr>
<td></td>
<td>The router memory is set too small. Therefore, the File Source cannot load any image into the PLC and enters the error state when attempting to do so. Enlarge the router memory.</td>
</tr>
</tbody>
</table>

Images from FS Control not in PLC

<table>
<thead>
<tr>
<th>Error pattern</th>
<th>Solution 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Images are selected in the File Source Control and the File Source function block in the PLC is in the ACQUISITION state. Nevertheless, no images can be retrieved via GetCurrentImage</td>
<td>BaseDir not set</td>
</tr>
<tr>
<td></td>
<td>The Read from Target button was actuated for the first time after TwinCAT was activated. In this case, the path from which the File Source images are read has not yet been transferred to the running TcCOM object. Therefore no images can be loaded, even though they are selected in the File Source Control. Reactivate TwinCAT after you have pressed Read from Target.</td>
</tr>
<tr>
<td></td>
<td>Trigger mode</td>
</tr>
<tr>
<td></td>
<td>The Trigger mode is selected in the File Source Control, but only the method GetCurrentImage is called in the PLC, not TriggerImage. Trigger an image by calling the method TriggerImage in the PLC, or deactivate the Trigger mode in the File Source Control.</td>
</tr>
</tbody>
</table>

9.3.3 Assistants

Solutions for situations in which problems occur with the operation of assistants are explained below.
9.3.3.1 Timeouts
A timeout error occurred during the communication with a camera. This can happen in different places as illustrated by the following cases.

### Situation

**During image display in the development environment:**

![Image](image1.png)

### Solution

In the case of timeout errors, two different types of timeout are essentially relevant: One is the transmission timeout, which limits the communication between the Image Acquisition module and the camera. The other is the ADS communication timeout, which limits the communication between the user interface and the Image Acquisition module.

Adjust the transmission timeout in the TcCOM parameters of the GVCP module in the configuration tree [75]. Also adjust the number of MaxTimeouts.

Make sure that, if necessary, you adjust the ADS communication timeout in the Settings [78], as this should be larger than the product of MaxTimeouts and Transmission Timeout.

Gradually increase the times and, if necessary, the number of attempts until no more error messages occur. Make sure, however, that you do not set the times and number of attempts unnecessarily high, as otherwise very long waiting times can occur.
9.3.3.2 Stalling image display

<table>
<thead>
<tr>
<th>Error pattern</th>
<th>The image display in the ADS Image Watch and camera assistants occasionally stalls.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solution 1</td>
<td><strong>Remote Desktop Protocol</strong></td>
</tr>
<tr>
<td></td>
<td>In individual cases it is possible that the Remote Desktop Protocol and the network card do not work well with the transfer from the TwinCAT Vision development environment or the image display tools.</td>
</tr>
<tr>
<td></td>
<td>Work either locally and connect yourself with the target in TwinCAT or work directly on the target computer.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Error pattern</th>
<th>Not all images are displayed in the ADS Image Watch and camera assistants.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solution 1</td>
<td><strong>Reduction of the data quantity</strong></td>
</tr>
<tr>
<td></td>
<td>As the images are retrieved from the controller by ADS in order to display them, it may be the case in particular with large images, high frame rates or several ADS Image Watch windows that not all image data can be retrieved.</td>
</tr>
<tr>
<td></td>
<td>Therefore reduce the number of images to be displayed simultaneously, reduce the frame rate or the data quantity per image by using ROIs.</td>
</tr>
</tbody>
</table>

9.3.3.3 Camera not found with Device Discovery

<table>
<thead>
<tr>
<th>Situation</th>
<th>You are in the process of creating a GigE Vision Camera object. You have successfully set up a TwinCAT RT-Ethernet adapter with IP stack. Nevertheless, the Device Discovery dialog remains empty and there is no error message.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solution 1</td>
<td><strong>Wrong network adapter selected</strong></td>
</tr>
<tr>
<td>Solution 2</td>
<td><strong>No GigE Vision camera</strong></td>
</tr>
<tr>
<td></td>
<td>A camera with a Gigabit Ethernet interface does not necessarily have to ‘speak’ GigE Vision. Please pay close attention to the manufacturer's instructions for the transmission protocol.</td>
</tr>
<tr>
<td>Solution 3</td>
<td><strong>Incompatible network adapter</strong></td>
</tr>
<tr>
<td></td>
<td>The network adapter used...</td>
</tr>
</tbody>
</table>
9.3.3.4 Error when loading camera objects

Error message

In configuration mode, the following Visual Studio message appears when the camera modules are reloaded or a camera is to be selected via the Discover Devices button in the Choose Target Camera dialog.

```
Microsoft Visual Studio

Init16:10: Set State TComObj SAFEOP OP: Set Objects (3) to OP >> AdsError: 1792 (0x700, "")
```

Solution

More cameras are connected than have been configured.

The default setting of the network adapter allows up to 2 cameras to be operated simultaneously. If you wish to operate more cameras, increase the Init parameters ipMaxReceiver and UdpMaxReceiver of the network adapter to twice the number of connected cameras (see Configuration of the network adapter [56]).

9.3.3.5 Warning: Reading/Writing is not permitted

Warning

This warning may appear when connecting a camera.

Solution

The warning indicates that a write or read command that TwinCAT Vision sends to the camera is not allowed. This can happen if the definition of the access rights on a register in the GenAPI description deviates from the actual behavior of the camera. Usually this camera register is unimportant for the user. To ascertain which register this concerns, you can search for the displayed register address in the Configuration Assistant [71].

9.3.4 TwinCAT HMI packages

Solutions for situations in which TwinCAT HMI packages do not work as expected are explained below.

9.3.4.1 Server extension

Images are not displayed

<table>
<thead>
<tr>
<th>Situation</th>
<th>Images from the PLC are not displayed in the HMI, although the server extension [1359] is installed and the image symbols are appropriately linked.</th>
</tr>
</thead>
</table>
| Solution 1| Missing administrator rights

You have chosen "Stream" as the image format, but you haven't started the HMI server as an administrator. Either select a different image format (BMP, PNG or JPG), or start the HMI server as an administrator.
### Images missing from a published page

<table>
<thead>
<tr>
<th>Situation</th>
<th>Solution 1</th>
<th>Solution 2</th>
<th>Solution 3</th>
<th>Solution 4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Situation</strong></td>
<td>The TcHmiVision server extension has been successfully integrated into the project. Images are displayed in the Live View or on the HMI provided by the Engineering server. The project could also be published error-free on the standalone TcHmi server. However, no images are displayed on the published HMI.</td>
<td><strong>Missing target licenses</strong></td>
<td><strong>Wrong configuration of the runtime environment</strong></td>
<td><strong>Restricted user rights</strong></td>
</tr>
<tr>
<td><strong>Solution 1</strong></td>
<td>A target license is included in the TC3 HMI server license. Additional licenses are required if the HMI server is to communicate with several targets (ADS connections).</td>
<td>The default configuration used in Engineering may differ from the published configuration (different AmsNetIds). Therefore, check the configuration of the runtime environment and adjust it if necessary.</td>
<td>The TwinCAT HMI user management is used and the currently logged-in user has no permissions to view the images or receive the extension data. Check the authorizations of your user groups.</td>
<td>As the data quantity can be quite large, especially in the case of several images that have to be retrieved simultaneously from the controller by ADS and converted by the HMI server, this can lead to bottlenecks. Several options are available to remedy this: reducing the size of the images in the controller, increasing the refresh rates of the individual images, using different refresh rates or using a compressed image format.</td>
</tr>
</tbody>
</table>

### 9.3.5 Camera communication

In case of problems with the connection, especially with sent or received frames, as well as problems with the parameterization or missing images, you will find the following diagnostic information in the network adapter and under the Image Acquisition Module of the camera object.

#### Diagnostic options in the RT Ethernet adapter

1. First select the RT Ethernet adapter used for the respective camera communication.

   ![Image](image)

   1. Open it by double-clicking and go to the **Statistics** tab.

   ![Image](image)

   Here you can see whether frames are generally sent and received or whether errors occur. Furthermore you can read the utilization of the connection in percent at **Recv Utilization**. A very high utilization of more than 98% already indicates an overload, especially if several cameras are operated on one adapter.

   Further diagnostic information can be found in the IpStack.
1. To do this, first open the underlying IpStack by double-clicking on it.
2. Go to the **Parameter (Online)** tab.
   ⇒ Here you can see the counter values for the individual categories.

### Diagnostic options in the Image Acquisition modules

1. First select the respective camera object.
   - [Camera1]
     - [Camera1 Image Acquisition (CGvImageAcquisition)]
       - [Gvcp Module (CGvcpProtocol)]
       - [Gvsp Module (CGvspProtocol)]
     - [Camera1 Image Acquisition Simulation (CGvImageAcquisitionSimulation)]
     - [Camera1 Ads Communicator (CGvAdsCommunicator)]
     - [Camera1 image Provider (CGvImageProvider)]

2. Open the **Parameters (online)** tab under the Gvcp module.
Here you get the information about the Control Protocol communication. This includes e.g. the heartbeat, reading and writing of registers and commands like Start Acquisition. If everything is OK, only the two nSuccess values should count up. If timeouts occur, refer to this chapter Timeouts [1480] for remedies. For other problems a deeper diagnosis with Wireshark is necessary.

The diagnostic information for the Streaming Protocol can be found under the Gvsp module on the Parameter (online) tab. These counters refer purely to the transmitted image data and are divided into the individual telegrams/packages and complete images/blocks.

If everything is Ok, only the nSuccess and the nCompletedBlocks value should count up.

The nResendRequests already indicate problems, but in this case they could still be answered, because all other counters are still zero. In case of a timeout, the BlockTimeout should be increased in the Configuration Assistant under the GVSP Module settings [97]. Counting up the nBlockIdOverflows is normal and does not represent an error. The counter will eventually overflow due to the value range or will be reset when the camera is restarted.

9.4 ADS Return Codes

Grouping of error codes: 0x000 [1486], 0x500 [1487], 0x700 [1488], 0x1000 [1490]

Global error codes
### Appendix

<table>
<thead>
<tr>
<th>Hex</th>
<th>Dec</th>
<th>HRESULT</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>0</td>
<td>0x9811 0000</td>
<td>ERR_NOERROR</td>
<td>No error.</td>
</tr>
<tr>
<td>0x1</td>
<td>1</td>
<td>0x9811 0001</td>
<td>ERR_INTERNAL</td>
<td>Internal error.</td>
</tr>
<tr>
<td>0x2</td>
<td>2</td>
<td>0x9811 0002</td>
<td>ERR_NORTIME</td>
<td>No real-time.</td>
</tr>
<tr>
<td>0x3</td>
<td>3</td>
<td>0x9811 0003</td>
<td>ERR_ALLOCLOCKEDMEM</td>
<td>Allocation locked – memory error.</td>
</tr>
<tr>
<td>0x4</td>
<td>4</td>
<td>0x9811 0004</td>
<td>ERR_INSERTMAILBOX</td>
<td>Mailbox full – the ADS message could not be sent. Reducing the number of ADS messages per cycle will help.</td>
</tr>
<tr>
<td>0x5</td>
<td>5</td>
<td>0x9811 0005</td>
<td>ERR_WRONGRECEIVEHMSG</td>
<td>Wrong HMSG.</td>
</tr>
<tr>
<td>0x6</td>
<td>6</td>
<td>0x9811 0006</td>
<td>ERR_TARGETPORTNOTFOUND</td>
<td>Target port not found – ADS server is not started or is not reachable.</td>
</tr>
<tr>
<td>0x7</td>
<td>7</td>
<td>0x9811 0007</td>
<td>ERR_TARGETMACHINENOTFOUND</td>
<td>Target computer not found – AMS route was not found.</td>
</tr>
<tr>
<td>0x8</td>
<td>8</td>
<td>0x9811 0008</td>
<td>ERR_UNKNOWNCMDID</td>
<td>Unknown command ID.</td>
</tr>
<tr>
<td>0x9</td>
<td>9</td>
<td>0x9811 0009</td>
<td>ERR_BADTASKID</td>
<td>Invalid task ID.</td>
</tr>
<tr>
<td>0xA</td>
<td>10</td>
<td>0x9811 000A</td>
<td>ERR_NOIO</td>
<td>No IO.</td>
</tr>
<tr>
<td>0xB</td>
<td>11</td>
<td>0x9811 000B</td>
<td>ERR_UNKNOWNAMSCMD</td>
<td>Unknown AMS command.</td>
</tr>
<tr>
<td>0xC</td>
<td>12</td>
<td>0x9811 000C</td>
<td>ERR_WIN32ERROR</td>
<td>Win32 error.</td>
</tr>
<tr>
<td>0xD</td>
<td>13</td>
<td>0x9811 000D</td>
<td>ERR_PORTNOTCONNECTED</td>
<td>Port not connected.</td>
</tr>
<tr>
<td>0xE</td>
<td>14</td>
<td>0x9811 000E</td>
<td>ERR_INVALIDAMSMEM</td>
<td>Invalid AMS length.</td>
</tr>
<tr>
<td>0xF</td>
<td>15</td>
<td>0x9811 000F</td>
<td>ERR_INVALIDAMSNETID</td>
<td>Invalid AMS Net ID.</td>
</tr>
<tr>
<td>0x0</td>
<td>16</td>
<td>0x9811 0010</td>
<td>ERR_LOWINSTLEVEL</td>
<td>Installation level is too low – TwinCAT 2 license error.</td>
</tr>
<tr>
<td>0x1</td>
<td>17</td>
<td>0x9811 0011</td>
<td>ERR_NODEBUGINTAVAILABLE</td>
<td>No debugging available.</td>
</tr>
<tr>
<td>0x2</td>
<td>18</td>
<td>0x9811 0012</td>
<td>ERR_PORTDISABLED</td>
<td>Port disabled – TwinCAT system service not started.</td>
</tr>
<tr>
<td>0x3</td>
<td>19</td>
<td>0x9811 0013</td>
<td>ERR_PORTALREADYCONNECTED</td>
<td>Port already connected.</td>
</tr>
<tr>
<td>0x4</td>
<td>20</td>
<td>0x9811 0014</td>
<td>ERR_AMSSYNC_W32ERROR</td>
<td>AMS Sync Win32 error.</td>
</tr>
<tr>
<td>0x5</td>
<td>21</td>
<td>0x9811 0015</td>
<td>ERR_AMSSYNC_TIMEOUT</td>
<td>AMS Sync Timeout.</td>
</tr>
<tr>
<td>0x6</td>
<td>22</td>
<td>0x9811 0016</td>
<td>ERR_AMSSYNC_AMSERROR</td>
<td>AMS Sync error.</td>
</tr>
<tr>
<td>0x7</td>
<td>23</td>
<td>0x9811 0017</td>
<td>ERR_AMSSYNC_NOINDEXINMAP</td>
<td>No index map for AMS Sync available.</td>
</tr>
<tr>
<td>0x8</td>
<td>24</td>
<td>0x9811 0018</td>
<td>ERR_INVALIDAMSPORT</td>
<td>Invalid AMS port.</td>
</tr>
<tr>
<td>0x9</td>
<td>25</td>
<td>0x9811 0019</td>
<td>ERR_NOMEMORY</td>
<td>No memory.</td>
</tr>
<tr>
<td>0xA</td>
<td>26</td>
<td>0x9811 001A</td>
<td>ERR_TCPSEND</td>
<td>TCP send error.</td>
</tr>
<tr>
<td>0xB</td>
<td>27</td>
<td>0x9811 001B</td>
<td>ERR_HOSTUNREACHABLE</td>
<td>Host unreachable.</td>
</tr>
<tr>
<td>0xC</td>
<td>28</td>
<td>0x9811 001C</td>
<td>ERR_INVALIDAMSFRAAGMENT</td>
<td>Invalid AMS fragment.</td>
</tr>
<tr>
<td>0xD</td>
<td>29</td>
<td>0x9811 001D</td>
<td>ERR_TLSSEND</td>
<td>TLS send error – secure ADS connection failed.</td>
</tr>
<tr>
<td>0xE</td>
<td>30</td>
<td>0x9811 001E</td>
<td>ERR_ACCESSDENIED</td>
<td>Access denied – secure ADS access denied.</td>
</tr>
</tbody>
</table>

### Router error codes

<table>
<thead>
<tr>
<th>Hex</th>
<th>Dec</th>
<th>HRESULT</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x500</td>
<td>1280</td>
<td>0x9811 0500</td>
<td>ROUTERERR_NOLOCKEDMEMORY</td>
<td>Locked memory cannot be allocated.</td>
</tr>
<tr>
<td>0x501</td>
<td>1281</td>
<td>0x9811 0501</td>
<td>ROUTERERR_RESIZEMEMORY</td>
<td>The router memory size could not be changed.</td>
</tr>
<tr>
<td>0x502</td>
<td>1282</td>
<td>0x9811 0502</td>
<td>ROUTERERR_MAILBOXFULL</td>
<td>The mailbox has reached the maximum number of possible messages.</td>
</tr>
<tr>
<td>0x503</td>
<td>1283</td>
<td>0x9811 0503</td>
<td>ROUTERERR_DEBUGBOXFULL</td>
<td>The Debug mailbox has reached the maximum number of possible messages.</td>
</tr>
<tr>
<td>0x504</td>
<td>1284</td>
<td>0x9811 0504</td>
<td>ROUTERERR_UNKNOWNPORTTYPE</td>
<td>The port type is unknown.</td>
</tr>
<tr>
<td>0x505</td>
<td>1285</td>
<td>0x9811 0505</td>
<td>ROUTERERR_NOTINITIALIZED</td>
<td>The router is not initialized.</td>
</tr>
<tr>
<td>0x506</td>
<td>1286</td>
<td>0x9811 0506</td>
<td>ROUTERERR_PORTALREADYINUSE</td>
<td>The port number is already assigned.</td>
</tr>
<tr>
<td>0x507</td>
<td>1287</td>
<td>0x9811 0507</td>
<td>ROUTERERR_NOTREGISTERED</td>
<td>The port is not registered.</td>
</tr>
<tr>
<td>0x508</td>
<td>1288</td>
<td>0x9811 0508</td>
<td>ROUTERERR_NOMOREQUEUES</td>
<td>The maximum number of ports has been reached.</td>
</tr>
<tr>
<td>0x509</td>
<td>1289</td>
<td>0x9811 0509</td>
<td>ROUTERERR_INVALIDPORT</td>
<td>The port is invalid.</td>
</tr>
<tr>
<td>0x50A</td>
<td>1290</td>
<td>0x9811 050A</td>
<td>ROUTERERR_NOTACTIVATED</td>
<td>The router is not active.</td>
</tr>
<tr>
<td>0x50B</td>
<td>1291</td>
<td>0x9811 050B</td>
<td>ROUTERERR_FRAGMENTBOXFULL</td>
<td>The mailbox has reached the maximum number for fragmented messages.</td>
</tr>
<tr>
<td>0x50C</td>
<td>1292</td>
<td>0x9811 050C</td>
<td>ROUTERERR_FRAGMENTTIMEOUT</td>
<td>A fragment timeout has occurred.</td>
</tr>
<tr>
<td>0x50D</td>
<td>1293</td>
<td>0x9811 050D</td>
<td>ROUTERERR_TOBEREMOVED</td>
<td>The port is removed.</td>
</tr>
</tbody>
</table>
Appendix

General ADS error codes
<table>
<thead>
<tr>
<th>Hex</th>
<th>Dec</th>
<th>HRESULT</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x700</td>
<td>1792</td>
<td>0x9811 0700</td>
<td>ADSERR_DEVICE_ERROR</td>
<td>General device error.</td>
</tr>
<tr>
<td>0x701</td>
<td>1793</td>
<td>0x9811 0701</td>
<td>ADSERR_DEVICE_SRVNOTSUPP</td>
<td>Service is not supported by the server.</td>
</tr>
<tr>
<td>0x702</td>
<td>1794</td>
<td>0x9811 0702</td>
<td>ADSERR_DEVICE_INVALIDGRP</td>
<td>Invalid index group.</td>
</tr>
<tr>
<td>0x703</td>
<td>1795</td>
<td>0x9811 0703</td>
<td>ADSERR_DEVICE_INVALIDOFFSET</td>
<td>Invalid index offset.</td>
</tr>
<tr>
<td>0x704</td>
<td>1796</td>
<td>0x9811 0704</td>
<td>ADSERR_DEVICE_INVALIDACCESS</td>
<td>Reading or writing not permitted.</td>
</tr>
<tr>
<td>0x705</td>
<td>1797</td>
<td>0x9811 0705</td>
<td>ADSERR_DEVICE_INVALIDSIZE</td>
<td>Parameter size not correct.</td>
</tr>
<tr>
<td>0x706</td>
<td>1798</td>
<td>0x9811 0706</td>
<td>ADSERR_DEVICE_INVALIDDATA</td>
<td>Invalid data values.</td>
</tr>
<tr>
<td>0x707</td>
<td>1799</td>
<td>0x9811 0707</td>
<td>ADSERR_DEVICE_NOTREADY</td>
<td>Device is not ready to operate.</td>
</tr>
<tr>
<td>0x708</td>
<td>1800</td>
<td>0x9811 0708</td>
<td>ADSERR_DEVICE_BUSY</td>
<td>Device is busy.</td>
</tr>
<tr>
<td>0x709</td>
<td>1801</td>
<td>0x9811 0709</td>
<td>ADSERR_DEVICE_INVALIDCONTEXT</td>
<td>Invalid operating system context. This can result from use of ADS function blocks in different tasks. It may be possible to resolve this through Multi-task data access synchronization in the PLC.</td>
</tr>
<tr>
<td>0x70A</td>
<td>1802</td>
<td>0x9811 070A</td>
<td>ADSERR_DEVICE_NOMEMORY</td>
<td>Insufficient memory.</td>
</tr>
<tr>
<td>0x70B</td>
<td>1803</td>
<td>0x9811 070B</td>
<td>ADSERR_DEVICE_INVALIDPARAM</td>
<td>Invalid parameter values.</td>
</tr>
<tr>
<td>0x70C</td>
<td>1804</td>
<td>0x9811 070C</td>
<td>ADSERR_DEVICE_NOTFOUND</td>
<td>Not found (files,...).</td>
</tr>
<tr>
<td>0x70D</td>
<td>1805</td>
<td>0x9811 070D</td>
<td>ADSERRDEVICE_SYNTAX</td>
<td>Syntax error in file or command.</td>
</tr>
<tr>
<td>0x70E</td>
<td>1806</td>
<td>0x9811 070E</td>
<td>ADSERRDEVICE_INCOMPATIBLE</td>
<td>Objects do not match.</td>
</tr>
<tr>
<td>0x70F</td>
<td>1807</td>
<td>0x9811 070F</td>
<td>ADSERRDEVICE_EXISTS</td>
<td>Object already exists.</td>
</tr>
<tr>
<td>0x710</td>
<td>1808</td>
<td>0x9811 0710</td>
<td>ADSERRDEVICE_SYMBOLNOTFOUND</td>
<td>Symbol not found.</td>
</tr>
<tr>
<td>0x711</td>
<td>1809</td>
<td>0x9811 0711</td>
<td>ADSERRDEVICE_SYMBOLVERSIONINVALID</td>
<td>Invalid symbol version. This can occur due to an online change. Create a new handle.</td>
</tr>
<tr>
<td>0x712</td>
<td>1810</td>
<td>0x9811 0712</td>
<td>ADSERRDEVICE_INVALIDSTATE</td>
<td>Device (server) is in invalid state.</td>
</tr>
<tr>
<td>0x713</td>
<td>1811</td>
<td>0x9811 0713</td>
<td>ADSERRDEVICE_TRANSMODENOTSUPP</td>
<td>AdsTransMode not supported.</td>
</tr>
<tr>
<td>0x714</td>
<td>1812</td>
<td>0x9811 0714</td>
<td>ADSERRDEVICE_NOTIFYVINDINVAILD</td>
<td>Notification handle is invalid.</td>
</tr>
<tr>
<td>0x715</td>
<td>1813</td>
<td>0x9811 0715</td>
<td>ADSERRDEVICE_CLIENTUNKNOWN</td>
<td>Notification client not registered.</td>
</tr>
<tr>
<td>0x716</td>
<td>1814</td>
<td>0x9811 0716</td>
<td>ADSERRDEVICE_NOMOREHDLS</td>
<td>No further handle available.</td>
</tr>
<tr>
<td>0x717</td>
<td>1815</td>
<td>0x9811 0717</td>
<td>ADSERRDEVICE_INVALIDWATCHSIZE</td>
<td>Notification size too large.</td>
</tr>
<tr>
<td>0x718</td>
<td>1816</td>
<td>0x9811 0718</td>
<td>ADSERRDEVICE_NOTINIT</td>
<td>Device not initialized.</td>
</tr>
<tr>
<td>0x719</td>
<td>1817</td>
<td>0x9811 0719</td>
<td>ADSERRDEVICE_TIMEOUT</td>
<td>Device has a timeout.</td>
</tr>
<tr>
<td>0x71A</td>
<td>1818</td>
<td>0x9811 071A</td>
<td>ADSERRDEVICE_NOINTERFACE</td>
<td>Interface query failed.</td>
</tr>
<tr>
<td>0x71B</td>
<td>1819</td>
<td>0x9811 071B</td>
<td>ADSERRDEVICE_INVALIDINTERFACE</td>
<td>Wrong interface requested.</td>
</tr>
<tr>
<td>0x71C</td>
<td>1820</td>
<td>0x9811 071C</td>
<td>ADSERRDEVICE_INVALIDCIDSID</td>
<td>Class ID is invalid.</td>
</tr>
<tr>
<td>0x71D</td>
<td>1821</td>
<td>0x9811 071D</td>
<td>ADSERRDEVICE_INVALIDOBJID</td>
<td>Object ID is invalid.</td>
</tr>
<tr>
<td>0x71E</td>
<td>1822</td>
<td>0x9811 071E</td>
<td>ADSERRDEVICE_PENDING</td>
<td>Request pending.</td>
</tr>
<tr>
<td>0x71F</td>
<td>1823</td>
<td>0x9811 071F</td>
<td>ADSERRDEVICE_ABORTED</td>
<td>Request is aborted.</td>
</tr>
<tr>
<td>0x720</td>
<td>1824</td>
<td>0x9811 0720</td>
<td>ADSERRDEVICE_WARNING</td>
<td>Signal warning.</td>
</tr>
<tr>
<td>0x721</td>
<td>1825</td>
<td>0x9811 0721</td>
<td>ADSERRDEVICE_INVALIDDARRAYIDX</td>
<td>Invalid array index.</td>
</tr>
<tr>
<td>0x722</td>
<td>1826</td>
<td>0x9811 0722</td>
<td>ADSERRDEVICE_SYMBOLNOTACTIVE</td>
<td>Symbol not active.</td>
</tr>
<tr>
<td>0x723</td>
<td>1827</td>
<td>0x9811 0723</td>
<td>ADSERRDEVICE_ACCESSDENIED</td>
<td>Access denied.</td>
</tr>
<tr>
<td>0x724</td>
<td>1828</td>
<td>0x9811 0724</td>
<td>ADSERRDEVICE_LICENSENOTFOUND</td>
<td>Missing license.</td>
</tr>
<tr>
<td>0x725</td>
<td>1829</td>
<td>0x9811 0725</td>
<td>ADSERRDEVICE_LICENSEEXPIRED</td>
<td>License expired.</td>
</tr>
<tr>
<td>0x726</td>
<td>1830</td>
<td>0x9811 0726</td>
<td>ADSERRDEVICE_LICENSEEXCEEDED</td>
<td>License exceeded.</td>
</tr>
<tr>
<td>0x727</td>
<td>1831</td>
<td>0x9811 0727</td>
<td>ADSERRDEVICE_LICENSEINVALID</td>
<td>Invalid license.</td>
</tr>
<tr>
<td>0x728</td>
<td>1832</td>
<td>0x9811 0728</td>
<td>ADSERRDEVICE_LICENSESYSTEMID</td>
<td>License problem: System ID is invalid.</td>
</tr>
<tr>
<td>0x729</td>
<td>1833</td>
<td>0x9811 0729</td>
<td>ADSERRDEVICE_LICENSENOTETIMELIMIT</td>
<td>License not limited in time.</td>
</tr>
<tr>
<td>0x72A</td>
<td>1834</td>
<td>0x9811 072A</td>
<td>ADSERRDEVICE_LICENSEFUTUREISSUE</td>
<td>License problem: Time in the future.</td>
</tr>
<tr>
<td>0x72B</td>
<td>1835</td>
<td>0x9811 072B</td>
<td>ADSERRDEVICE_LICENSESETIMETOLONG</td>
<td>License period too long.</td>
</tr>
<tr>
<td>0x72C</td>
<td>1836</td>
<td>0x9811 072C</td>
<td>ADSERRDEVICE_EXCEPTION</td>
<td>Exception at system startup.</td>
</tr>
<tr>
<td>0x72D</td>
<td>1837</td>
<td>0x9811 072D</td>
<td>ADSERRDEVICE_LICENSEDEPLICATED</td>
<td>License file read twice.</td>
</tr>
<tr>
<td>0x72E</td>
<td>1838</td>
<td>0x9811 072E</td>
<td>ADSERRDEVICE_LICENSEVALID</td>
<td>Invalid signature.</td>
</tr>
<tr>
<td>0x72F</td>
<td>1839</td>
<td>0x9811 072F</td>
<td>ADSERRDEVICE_CERTIFICATEINVALID</td>
<td>Invalid certificate.</td>
</tr>
<tr>
<td>0x730</td>
<td>1840</td>
<td>0x9811 0730</td>
<td>ADSERRDEVICE_LICENSEOEMNOTFOUND</td>
<td>Public key not known from OEM.</td>
</tr>
<tr>
<td>0x731</td>
<td>1841</td>
<td>0x9811 0731</td>
<td>ADSERRDEVICE_LICENSERESTRICTED</td>
<td>License not valid for this system ID.</td>
</tr>
<tr>
<td>0x732</td>
<td>1842</td>
<td>0x9811 0732</td>
<td>ADSERRDEVICE_LICENSEDEMODENIED</td>
<td>Demo license prohibited.</td>
</tr>
<tr>
<td>0x733</td>
<td>1843</td>
<td>0x9811 0733</td>
<td>ADSERRDEVICE_INVALIDFCID</td>
<td>Invalid function ID.</td>
</tr>
<tr>
<td>0x734</td>
<td>1844</td>
<td>0x9811 0734</td>
<td>ADSERRDEVICE_OUTOF RANGE</td>
<td>Outside the valid range.</td>
</tr>
<tr>
<td>0x735</td>
<td>1845</td>
<td>0x9811 0735</td>
<td>ADSERRDEVICE_INVALIDALIGNMENT</td>
<td>Invalid alignment.</td>
</tr>
</tbody>
</table>
Appendix

<table>
<thead>
<tr>
<th>Hex</th>
<th>Dec</th>
<th>HRESULT</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x736</td>
<td>1846</td>
<td>0x9811 0736</td>
<td>ADSERR_DEVICE_LICENSEPLATFORM</td>
<td>Invalid platform level.</td>
</tr>
<tr>
<td>0x737</td>
<td>1847</td>
<td>0x9811 0737</td>
<td>ADSERR_DEVICE_FORWARD_PL</td>
<td>Context – forward to passive level.</td>
</tr>
<tr>
<td>0x738</td>
<td>1848</td>
<td>0x9811 0738</td>
<td>ADSERR_DEVICE_FORWARD_DL</td>
<td>Context – forward to dispatch level.</td>
</tr>
<tr>
<td>0x739</td>
<td>1849</td>
<td>0x9811 0739</td>
<td>ADSERR_DEVICE_FORWARD_RT</td>
<td>Context – forward to real-time.</td>
</tr>
<tr>
<td>0x740</td>
<td>1856</td>
<td>0x9811 0740</td>
<td>ADSERR_CLIENT_ERROR</td>
<td>Client error.</td>
</tr>
<tr>
<td>0x741</td>
<td>1857</td>
<td>0x9811 0741</td>
<td>ADSERR_CLIENT_INVALIDPARAM</td>
<td>Service contains an invalid parameter.</td>
</tr>
<tr>
<td>0x742</td>
<td>1858</td>
<td>0x9811 0742</td>
<td>ADSERR_CLIENT_LISTEMPTY</td>
<td>Polling list is empty.</td>
</tr>
<tr>
<td>0x743</td>
<td>1859</td>
<td>0x9811 0743</td>
<td>ADSERR_CLIENT_VARUSED</td>
<td>Var connection already in use.</td>
</tr>
<tr>
<td>0x744</td>
<td>1860</td>
<td>0x9811 0744</td>
<td>ADSERR_CLIENT_DUPLINVOKEID</td>
<td>The called ID is already in use.</td>
</tr>
<tr>
<td>0x745</td>
<td>1861</td>
<td>0x9811 0745</td>
<td>ADSERR_CLIENT_SYNC_TIMEOUT</td>
<td>Timeout has occurred – the remote terminal is not responding in the specified ADS timeout. The route setting of the remote terminal may be configured incorrectly.</td>
</tr>
<tr>
<td>0x746</td>
<td>1862</td>
<td>0x9811 0746</td>
<td>ADSERR_CLIENT_W32ERROR</td>
<td>Error in Win32 subsystem.</td>
</tr>
<tr>
<td>0x747</td>
<td>1863</td>
<td>0x9811 0747</td>
<td>ADSERR_CLIENT_TIMEOUT_INVALID</td>
<td>Invalid client timeout value.</td>
</tr>
<tr>
<td>0x748</td>
<td>1864</td>
<td>0x9811 0748</td>
<td>ADSERR_CLIENT_PORTNOTOPEN</td>
<td>Port not open.</td>
</tr>
<tr>
<td>0x749</td>
<td>1865</td>
<td>0x9811 0749</td>
<td>ADSERR_CLIENT_NOAMSSADR</td>
<td>No AMS address.</td>
</tr>
<tr>
<td>0x750</td>
<td>1872</td>
<td>0x9811 0750</td>
<td>ADSERR_CLIENT_SYNCINTERNAL</td>
<td>Internal error in Ads sync.</td>
</tr>
<tr>
<td>0x751</td>
<td>1873</td>
<td>0x9811 0751</td>
<td>ADSERR_CLIENT_ADDHASH</td>
<td>Hash table overflow.</td>
</tr>
<tr>
<td>0x752</td>
<td>1874</td>
<td>0x9811 0752</td>
<td>ADSERR_CLIENT_REMOVEHASH</td>
<td>Key not found in the table.</td>
</tr>
<tr>
<td>0x753</td>
<td>1875</td>
<td>0x9811 0753</td>
<td>ADSERR_CLIENT_NOMORESYM</td>
<td>No symbols in the cache.</td>
</tr>
<tr>
<td>0x754</td>
<td>1876</td>
<td>0x9811 0754</td>
<td>ADSERR_CLIENT_SYNCREINVALID</td>
<td>Invalid response received.</td>
</tr>
<tr>
<td>0x755</td>
<td>1877</td>
<td>0x9811 0755</td>
<td>ADSERR_CLIENT_SYNCPORLOCKED</td>
<td>Sync Port is locked.</td>
</tr>
</tbody>
</table>

RTime error codes

<table>
<thead>
<tr>
<th>Hex</th>
<th>Dec</th>
<th>HRESULT</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1000</td>
<td>4096</td>
<td>0x9811 1000</td>
<td>RTERR_INTERNAL</td>
<td>Internal error in the real-time system.</td>
</tr>
<tr>
<td>0x1001</td>
<td>4097</td>
<td>0x9811 1001</td>
<td>RTERR_BAD_TIMER_PERIODS</td>
<td>Timer value is not valid.</td>
</tr>
<tr>
<td>0x1002</td>
<td>4098</td>
<td>0x9811 1002</td>
<td>RTERR_INVALID_TASK_PTR</td>
<td>Task pointer has the invalid value 0 (zero).</td>
</tr>
<tr>
<td>0x1003</td>
<td>4099</td>
<td>0x9811 1003</td>
<td>RTERR_INVALID_STACK_PTR</td>
<td>Stack pointer has the invalid value 0 (zero).</td>
</tr>
<tr>
<td>0x1004</td>
<td>4100</td>
<td>0x9811 1004</td>
<td>RTERR_PRIO_EXISTS</td>
<td>The request task priority is already assigned.</td>
</tr>
<tr>
<td>0x1005</td>
<td>4101</td>
<td>0x9811 1005</td>
<td>RTERR_NOMORETCB</td>
<td>No free TCB (Task Control Block) available. The maximum number of TCBs is 64.</td>
</tr>
<tr>
<td>0x1006</td>
<td>4102</td>
<td>0x9811 1006</td>
<td>RTERR_NOMORESEMAS</td>
<td>No free semaphores available. The maximum number of semaphores is 64.</td>
</tr>
<tr>
<td>0x1007</td>
<td>4103</td>
<td>0x9811 1007</td>
<td>RTERR_NOMOREQUEUES</td>
<td>No free space available in the queue. The maximum number of positions in the queue is 64.</td>
</tr>
<tr>
<td>0x1008</td>
<td>4109</td>
<td>0x9811 1008</td>
<td>RTERR_EXTERNALALREADYDEF</td>
<td>An external synchronization interrupt is already applied.</td>
</tr>
<tr>
<td>0x100E</td>
<td>4110</td>
<td>0x9811 100E</td>
<td>RTERR_EXTERNALNOTDEFINED</td>
<td>No external sync interrupt applied.</td>
</tr>
<tr>
<td>0x100F</td>
<td>4111</td>
<td>0x9811 100F</td>
<td>RTERR_EXTERNALINSTALLFAILED</td>
<td>Application of the external synchronization interrupt has failed.</td>
</tr>
<tr>
<td>0x1010</td>
<td>4112</td>
<td>0x9811 1010</td>
<td>RTERR_IRQLNOTLESSOREQUAL</td>
<td>Call of a service function in the wrong context</td>
</tr>
<tr>
<td>0x1017</td>
<td>4119</td>
<td>0x9811 1017</td>
<td>RTERR_VMXNOT_SUPPORTED</td>
<td>Intel VT-x extension is not supported.</td>
</tr>
<tr>
<td>0x1018</td>
<td>4120</td>
<td>0x9811 1018</td>
<td>RTERR_VMX_DISABLED</td>
<td>Intel VT-x extension is not enabled in the BIOS.</td>
</tr>
<tr>
<td>0x1019</td>
<td>4121</td>
<td>0x9811 1019</td>
<td>RTERR_VMXCONTRLSMISSING</td>
<td>Missing function in Intel VT-x extension.</td>
</tr>
<tr>
<td>0x101A</td>
<td>4122</td>
<td>0x9811 101A</td>
<td>RTERR_VMXENABLEFAIL</td>
<td>Activation of Intel VT-x fails.</td>
</tr>
</tbody>
</table>

TCP Winsock error codes

<table>
<thead>
<tr>
<th>Hex</th>
<th>Dec</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x274C</td>
<td>10060</td>
<td>WSAETIMEOUT</td>
<td>A connection timeout has occurred - error while establishing the connection, because the remote terminal did not respond properly after a certain period of time, or the established connection could not be maintained because the connected host did not respond.</td>
</tr>
<tr>
<td>0x274D</td>
<td>10061</td>
<td>WSADERROR</td>
<td>Connection refused - no connection could be established because the target computer has explicitly rejected it. This error usually results from an attempt to connect to a service that is inactive on the external host, that is, a service for which no server application is running.</td>
</tr>
<tr>
<td>0x2751</td>
<td>10065</td>
<td>WS_EPSNOREACH</td>
<td>No route to host - a socket operation referred to an unavailable host.</td>
</tr>
</tbody>
</table>

More Winsock error codes: Win32 error codes
9.5 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: https://www.beckhoff.com

You will also find further documentation for Beckhoff components there.

Beckhoff Support

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

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

Hotline: +49 5246 963 157
Fax: +49 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 5246 963 460
Fax: +49 5246 963 479
e-mail: service@beckhoff.com

Beckhoff Headquarters

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

Phone: +49 5246 963 0
Fax: +49 5246 963 198
e-mail: info@beckhoff.com
web: https://www.beckhoff.com
9.6

This software contains third-party components. Please refer to the license file provided in the following folder for further information:

<TwinCatInstallPath>\3.1\System\Legal\TF7xxx Additional license information.txt
More Information:
www.beckhoff.com/twincat-vision/