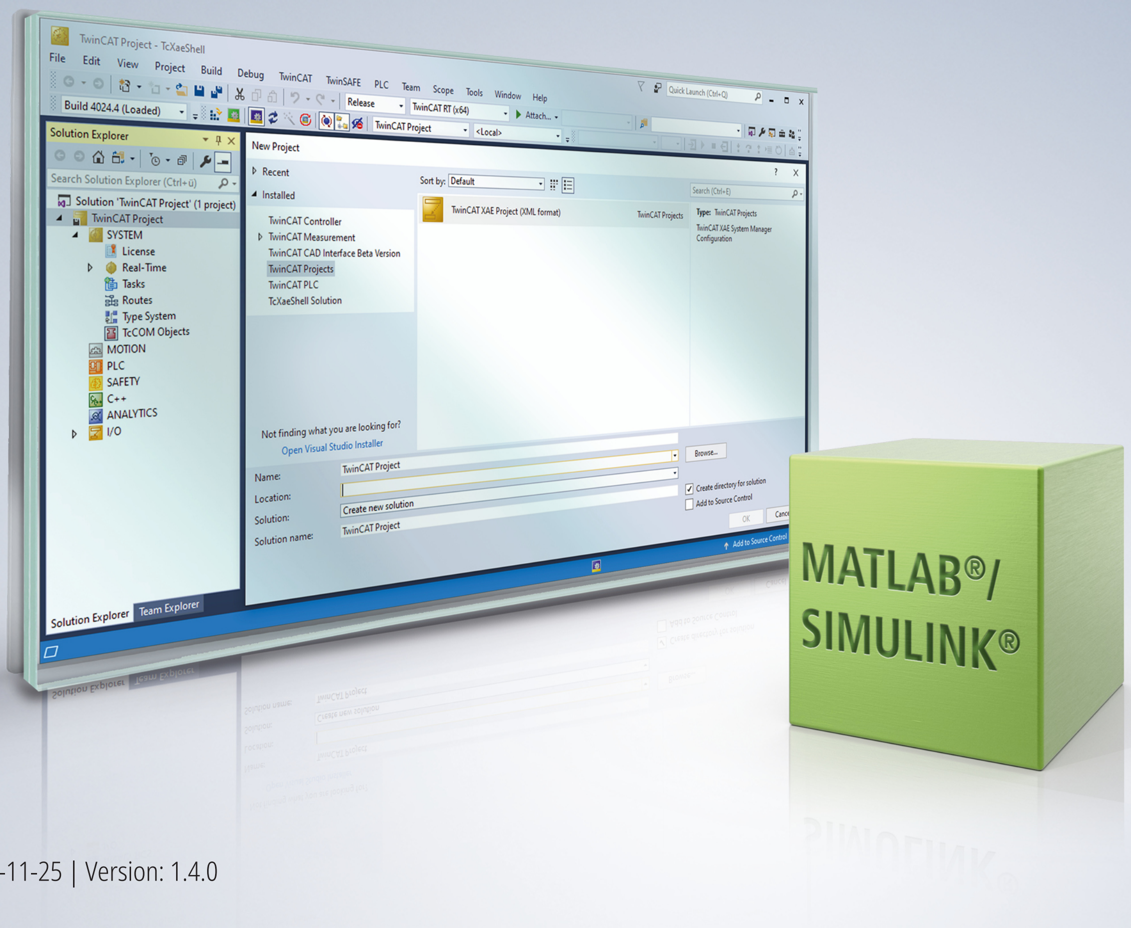
BECKHOFF New Automation Technology

Manual | EN

TwinCAT 3

MATLAB®/Simulink®



2024-11-25 | Version: 1.4.0

Table of contents

1 Foreword

1.1 Notes on the documentation

This description is intended exclusively for trained specialists in control and automation technology who are familiar with the applicable national standards.
For installation and commissioning of the components, it is absolutely necessary to observe the documentation and the following notes and explanations.
The qualified personnel is obliged to always use the currently valid documentation.

The responsible staff must ensure that the application or use of the products described satisfies all 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 notice.
No claims to modify 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 and licensed trademarks of Beckhoff Automation GmbH.
If third parties make use of designations or trademarks used in this publication for their own purposes, this could infringe upon the rights of the owners of the said designations.

Patents

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

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

Copyright

1.2 For your safety

Safety regulations

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

Exclusion of liability

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

Personnel qualification

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

Signal words

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

Personal injury warnings

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

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

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

Warning of damage to property or environment

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

Information on handling the product

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

1.3 Notes on information security

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

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

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

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

1.4 Documentation issue status

Version	Modifications
1.4.x	NEW: • View the version number of the block diagram. • For information on installing TwinCAT 3.1 Build 4026, see Block diagram [▶ 11].

2 Overview

MATLAB® and Simulink®

MATLAB® and Simulink® have become globally established environments for a wide variety of applications, even among budding engineers. There are many reasons for this development. MATLAB® and Simulink® provide environments where you can focus on the engineering task. This is perfect for didactic concepts in teaching environments and efficient in industrial applications. MATLAB®, with its numerous toolboxes, provides an ideal environment for developing algorithms and analyzing data. MATLAB® offers numerous functions for easy access to different data formats and harmonizes with the different data logging mechanisms of TwinCAT. Simulink® is focused on integrated support for Model-Based Design (MBD). This involves development, testing and verification based on a system model. The subsequent, automatic code generation for platforms such as TwinCAT represents an optimal solution for applying the tested code in production. Simulink® provides all the resources required for modeling multi-physics simulations and generating algorithms for controlling, regulating and AI. As a result, only high-quality codes tested on models are used on controllers.

Technologies and products for TwinCAT 3

TE1400 TwinCAT 3 Target for Simulink®

With the TwinCAT 3 Target for Simulink® it is possible to make models developed in Simulink® usable in TwinCAT 3. Various toolboxes, e.g. SimScape™, Stateflow™ or DSP System Toolbox™ can be integrated in Simulink®. Embedded MATLAB® function blocks are also supported. The models are automatically compiled in C/C++ code with the aid of the Simulink Coder™ and transformed into real-time capable TwinCAT objects with TwinCAT 3 Target for Simulink®. TwinCAT objects created from Simulink® have the same interfaces and properties as all other TwinCAT objects. They can be used fully in TwinCAT 3 Engineering, e.g. expand to a complete project with a PLC source code, debug and link with fieldbus devices, see the documentation TE1400 TwinCAT 3 Target for Simulink®.

TE1401 TwinCAT 3 Target for MATLAB®

TwinCAT 3 Target for MATLAB® enables MATLAB® functions to be used in TwinCAT 3. The functions are automatically transferred to TwinCAT objects and used seamlessly in TwinCAT 3 Engineering. The automatically generated modules can be integrated into the TwinCAT solution on the one hand as a TcCOM object and on the other hand as a PLC function block, also as part of a specially created PLC library. The inserted modules are downloaded with the complete TwinCAT project into the TwinCAT 3 runtime, where they are executed within the real-time environment like all other objects, see the documentation TE1401 TwinCAT 3 Target for MATLAB®.

TE1410 TwinCAT 3 Interface for MATLAB®/Simulink®

TwinCAT 3 Interface for MATLAB®/Simulink® establishes high-performance bidirectional communication between the TwinCAT runtime and MATLAB® or Simulink®. You can even use it for software-in-the-loop simulation with Simulink® during the system engineering phase. During the machine runtime you realize with this product for example:

- A MATLAB® app-based human-machine interface (HMI)
- A MATLAB® compiler runtime based Compute Server for e.g. predictive maintenance or parameter optimization

To the documentation of the TwinCAT 3 Interface for MATLAB®/Simulink®

TwinCAT basic technologies for MATLAB® and Simulink®

The tools presented in the following are free of license costs and are part of the TwinCAT 3 XAE or Full Setup.

TwinCAT 3 Block Diagram

The Block Diagram Viewer enables the display, debugging and parameterization of a TcCOM module via a block diagram displayed in the TwinCAT Engineering System. This tab is only available for TcCOM modules generated with Target for MATLAB® or Target for Simulink®. The export of the block diagram is optional for both targets.

Block diagram [▶ 11]

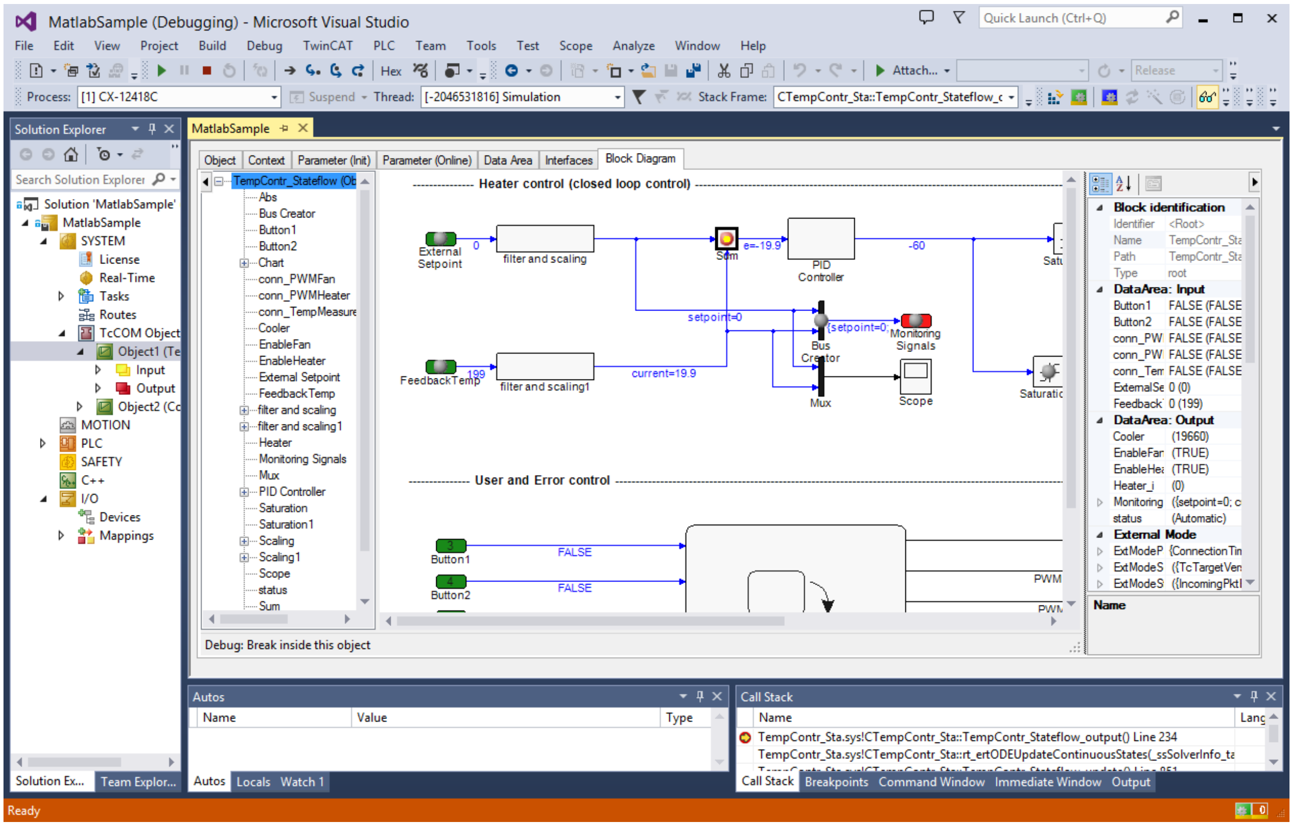
TwinCAT 3 Automation Interface

The TwinCAT 3 Automation Interface is a programming interface for TwinCAT projects, i.e. you can create a configuration in the System Manager, create a PLC and insert code, and instantiate TcCOM objects without opening TwinCAT via Visual Studio or the TwinCAT XAE. This API is also available for MATLAB®, among others.

TwinCAT Automation Interface: use in MATLAB® [▶ 26]

3 Block diagram

When generating a TwinCAT object from MATLAB® or Simulink®, the block diagram (Target for Simulink®) or the MATLAB® code (Target for MATLAB®) can optionally be exported as well. In this case, the block diagram or the MATLAB® code can be displayed in the TwinCAT development environment under the **Block Diagram** tab of the TcCOM instance:



Using this control, you can navigate through the entire structure of the block diagram or view the underlying MATLAB® code, and also view and change parameter values, display signal values and curves, and debug through the module using breakpoints in debugging mode. The control is designed such that it can also be used in a separate visualization.

The version number of the block diagram can be viewed by right-clicking in the control and selecting **About TC3 BlockDiagram**. Alternatively, you can view the version number in the Windows **Control Panel** under **Programs and Features** (entry: Beckhoff TwinCAT 3 BlockDiagram).

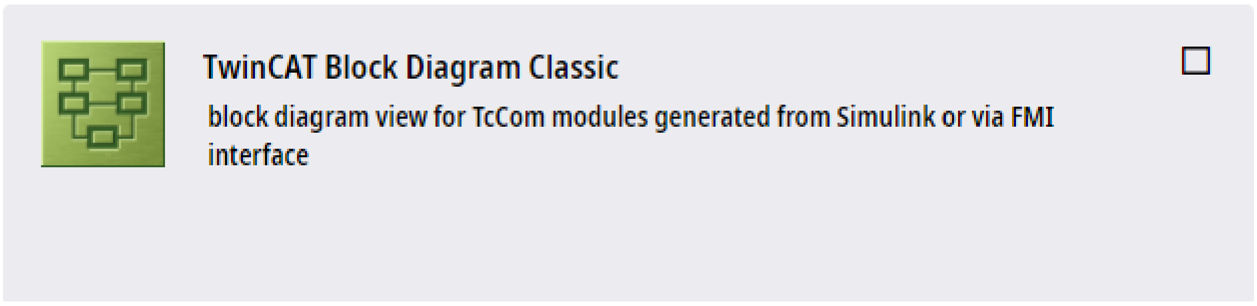
Installation

TwinCAT 3.1 Build 4024

The TwinCAT Block Diagram Setup is executed with the TwinCAT 3 XAE or Full Setup and does not have to be executed separately. The TwinCAT Block Diagram Setup is also supplied via the TE14xx TwinCAT Tools for MATLAB®.

TwinCAT 3.1 Build 4026

Install the following workload via the TwinCAT Package Manager:



TwinCAT Block Diagram Classic

block diagram view for TcCom modules generated from Simulink or via FMI interface

Command line: `tcpkg install TwinCAT.XAE.BlockDiagramClassic`

3.1 Simulink®-TcCOM

If a TwinCAT object was created with the TwinCAT Target for Simulink® and the block diagram export was executed in the process, the block diagram of the Simulink® model can be displayed as a control in the TwinCAT XAE.

3.1.1 Using the block diagram

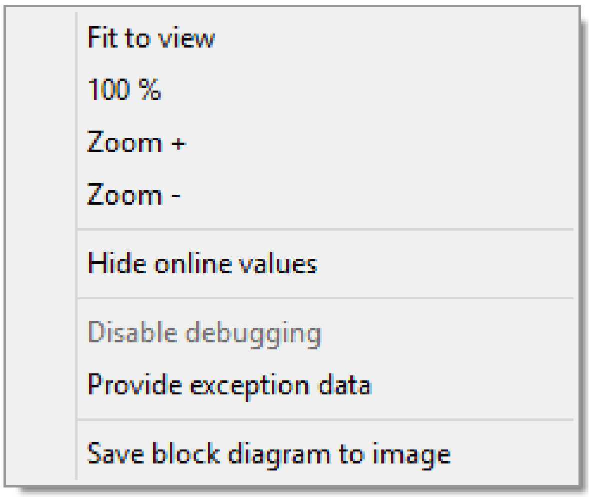
The block diagram export can be configured during generation of a TcCOM module from MATLAB® or Simulink®. If the export was enabled, the block diagram can be found in the TwinCAT development environment under the "Block Diagram" tab of the module instance.

Using shortcuts, drag & drop and a context menu you can navigate through the hierarchy of the TcCOM module, view parameter values, display signals values and obtain optional additional debug information.

Shortcut functions:

Shortcut	Function
Space	Zoom to current size of the block diagram tab
Backspace	Switch to the next higher hierarchical level
ESC	Switch to the next higher hierarchical level
CTRL + "+"	Zoom in
CTRL + "-"	Zoom out
F5	Attach Debugger (*System- > Real-Time -> C++ Debugger -> Enable C++ Debugger* must be activated)

Context menu functions:

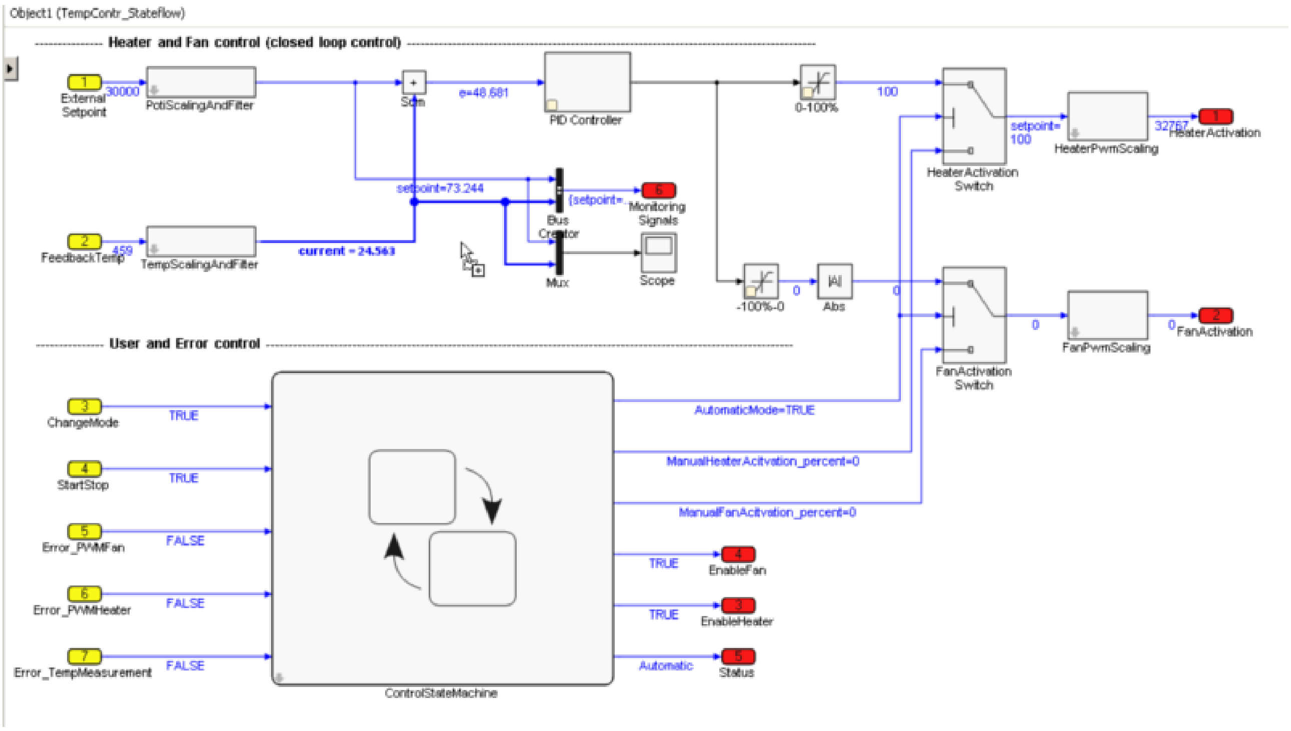


3.1.2 Display signal curves

For verification and troubleshooting it is often helpful to display signal curves. The block diagram offers the following options:

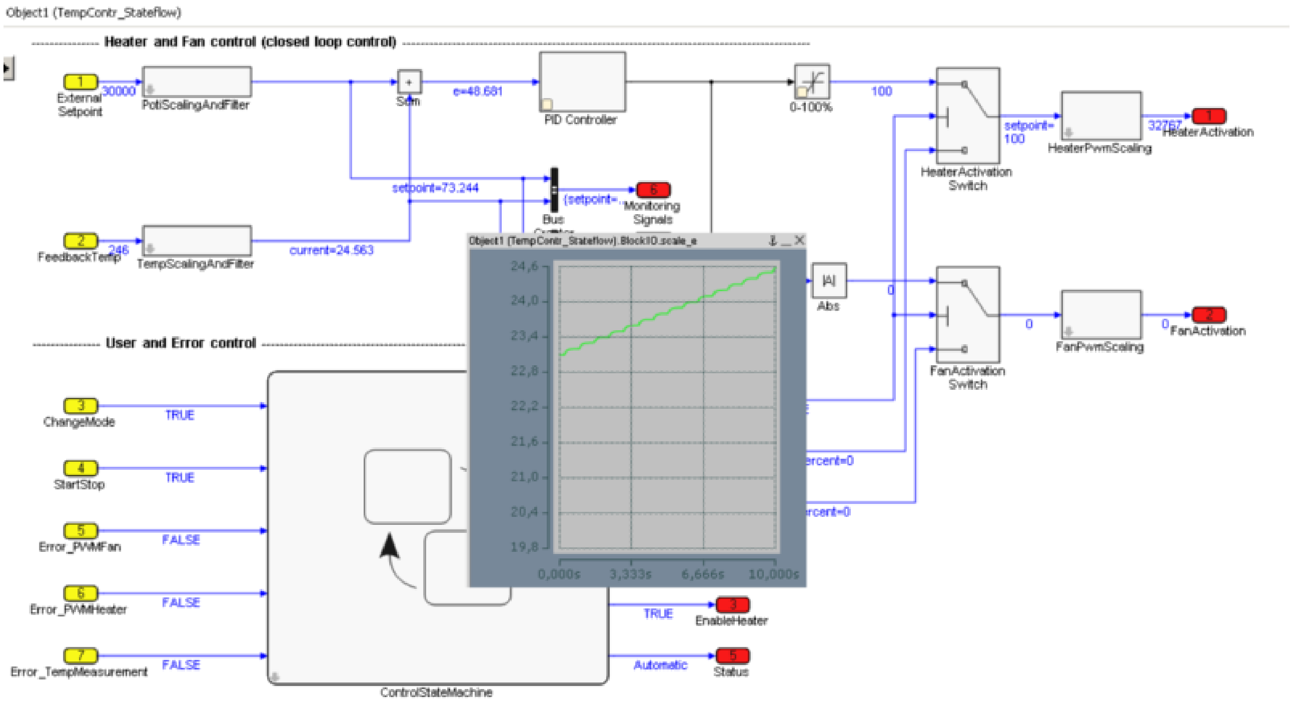
Display signal curves in the block diagram

The block diagram offers an option to display signal curves in a window. To this end, drag and drop a signal or block into a free area of the block diagram.



Create a scope in the block diagram

After the drop, a scope window opens in the block diagram.


Object1 (TempContr_Stateflow)
Heater and Fan control (closed loop control)
Object1 (TempContr_Stateflow).BlockIO.scale_e
24,6
24,0
23,4
22,8
22,2
21,6
21,0
20,4
19,8
0,000s
3,333s
6,666s
10,000s
ControlStateMachine

Display the scope in the block diagram

The title bar of the scope window offers the following options:

✖	Close window
⚓	Keep window in the foreground across all block diagram hierarchies
—	Minimize window to the title bar

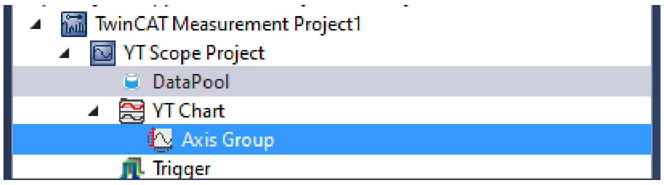
Displaying the scope in the block diagram control [▶ 24] requires a Scope View Professional (TE1300) license. No Scope View Professional license is required in TwinCAT XAE.

When creating a scope window in the block diagram for a Simulink® bus, all signals of the bus are directly displayed in the scope window.

The scope window in the block diagram can be used for a quick overview. For more detailed analyzes, it is advisable to analyze the signals in a TwinCAT Measurement project.

Display signal curves in TwinCAT 3 Scope

If the drop is not made to the block diagram control but to an Axis Group in a TwinCAT Measurement project, the signal is added there.



Add a signal in a TwinCAT 3 Scope

3.1.3 Module parameterization in the block diagram

To parameterize a TcCOM instance, the **parameter window** can be used directly in the block diagram. In addition, the Property table can be used, which can be expanded or collapsed on the right-hand edge of the block diagram. A basic distinction is made between different parameter values:

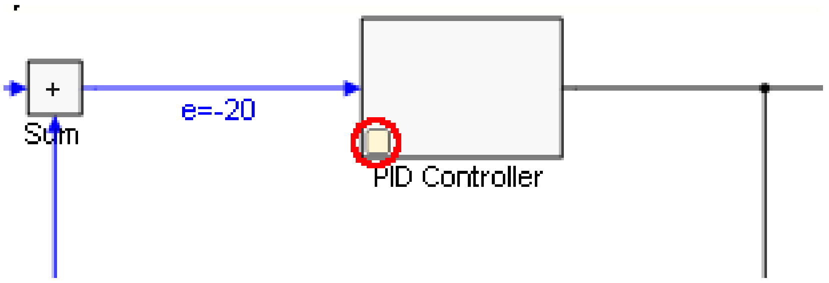
"Default", "Startup", "Online" and "Prepared"

The following value types can be found in the drop-down menu of the Property table of the block diagram:

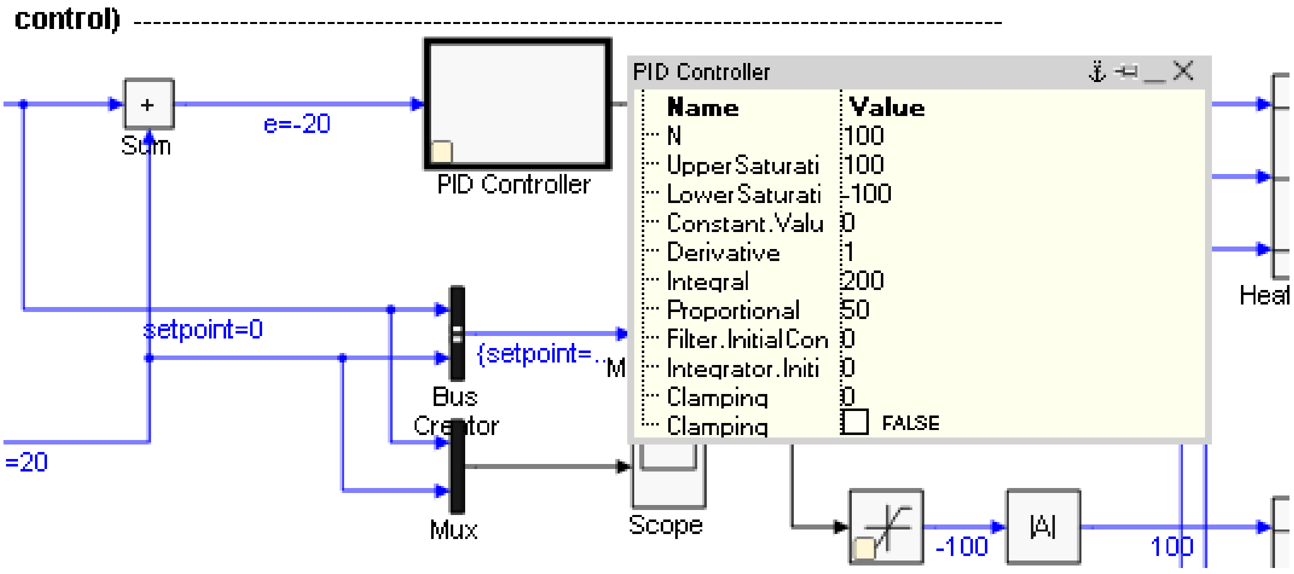
- **Default values** are the parameter values during code generation. They are invariably stored in the module description file and enable the manufacturing settings to be restored after parameter changes.
- **Startup values** are stored in the TwinCAT project file and downloaded to the module instance as soon as TwinCAT starts the module instance.
  Startup values for the input process image can also be specified in Simulink® modules. This allows the module to be started with non-zero input values, without the need for linking the inputs with other process images. Internal signals and output signals have no starting values, since they would, in any case, be overwritten in the first cycle.
- **Online values** are only available if the module was started on the target system. They show the current parameter value in the running module. This value can also be changed during runtime. Although in this case the corresponding input field has to be enabled via the context menu, in order to prevent accidental inputs.
- **Prepared values** can be specified whenever online values are available. They can be used to save various values, in order to write them consistently to the module. If prepared values have been specified, they are displayed in a table below the block diagram. The buttons to the right of the list can be used to download prepared values as online values and/or save them as starting value, or delete them.

Parameterization in the block diagram

Parameterizable blocks are marked with a yellow box in the block diagram.



Double-clicking on the block or a single click on the yellow box brings up a window with the parameters that can be changed.



Name	Value
N	100
UpperSaturati	100
LowerSaturati	-100
Constant.Valu	0
Derivative	1
Integral	200
Proportional	50
Filter.InitialCon	0
Integrator.Initi	0
Clamping	0
Clamping	☐ FALSE

If a value is changed, it can be applied with the following keyboard commands:

CTRL + Enter	Set online value directly
SHIFT + Enter	Set startup value
Enter	Set prepared value

The icons in the title bar have the following functions:

✕	Close window
⚓	Keep window in the foreground across all block diagram hierarchical levels
📌	Keep window open at the current block diagram hierarchical level
—	Minimize window to title bar

3.1.4 Debug

Different ways are available to find errors within a TcCOM module created with MATLAB®/Simulink®, or to analyze the behavior of the module within the overall architecture of the TwinCAT project.

Debugging in the block diagram

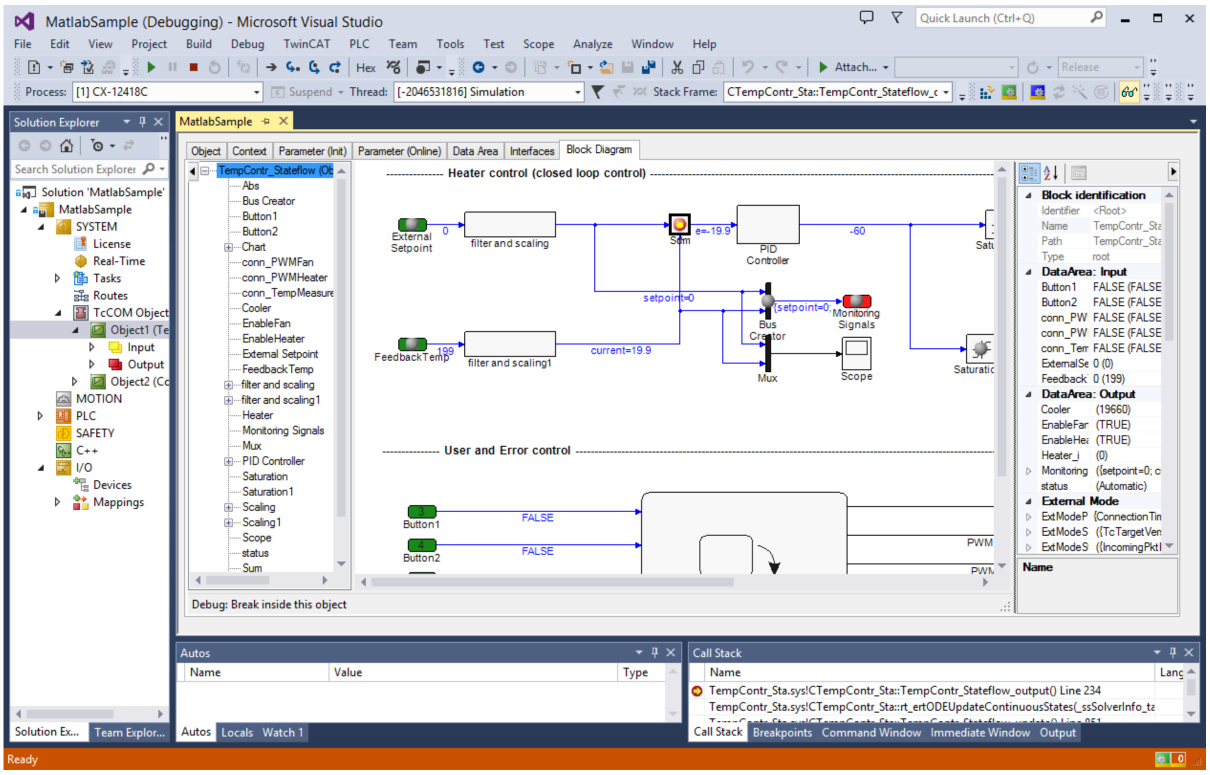
If the block diagram was exported during generation of the TcCOM module, it can be displayed in the TwinCAT development environment and used for debugging within the corresponding module instance, for example. To do so, the block diagram uses the Microsoft Visual Studio debugger, which can be linked with the TwinCAT runtime via the TwinCAT debugger port. Attach the debugger as described in the C++ section under Debugging.

Prerequisites for debugging within the block diagram are:

- The C/C++ source code of the TcCOM module must be present on the engineering systems, and the Visual Studio debugger must be able to find it. Ideally, debugging should take place on the system on which the code was generated. If the module was created on another system, the associated C/C++ source code can usually be made known by integrating the Visual Studio project into the TwinCAT C++ section. The file <ModelName>.vcxproj is located in the build directory, see Which files are created automatically during code generation and publishing?
- The module must have been created with the **Debug** configuration. When publishing takes place directly after the code generation, select the **Debug** setting in the Publish mechanism section under **publish configuration**. When publishing the module from the C++ section in TwinCAT, the debugger in the C++ node of the solution must be enabled; see C/C++ documentation, Debugging.
- During code generation, the options **Export block diagram** and **Export block diagram debug information** must be enabled in the coder settings under **Tc Advanced**.
- In the TwinCAT project, the debugger port must be enabled, as described in TwinCAT 3 C++ Enable C++ debugger.

Setting breakpoints in the block diagram

1. After attaching the debugger to the TwinCAT runtime, the possible breakpoints are assigned to the blocks in the block diagram and represented as points. Clicking on the desired breakpoint activates it, so that execution of the module instance is stopped next time the associated function block is executed. The color of the point provides information about the current state of the breakpoint:
   - Gray: breakpoint inactive
   - Red: breakpoint active. The program is stopped next time this function block is executed
   - Yellow dot in the middle: breakpoint hit. Program execution is currently stopped at this point
   - Blue dot in the middle: breakpoint hit (as yellow), but in a different instance of the module.

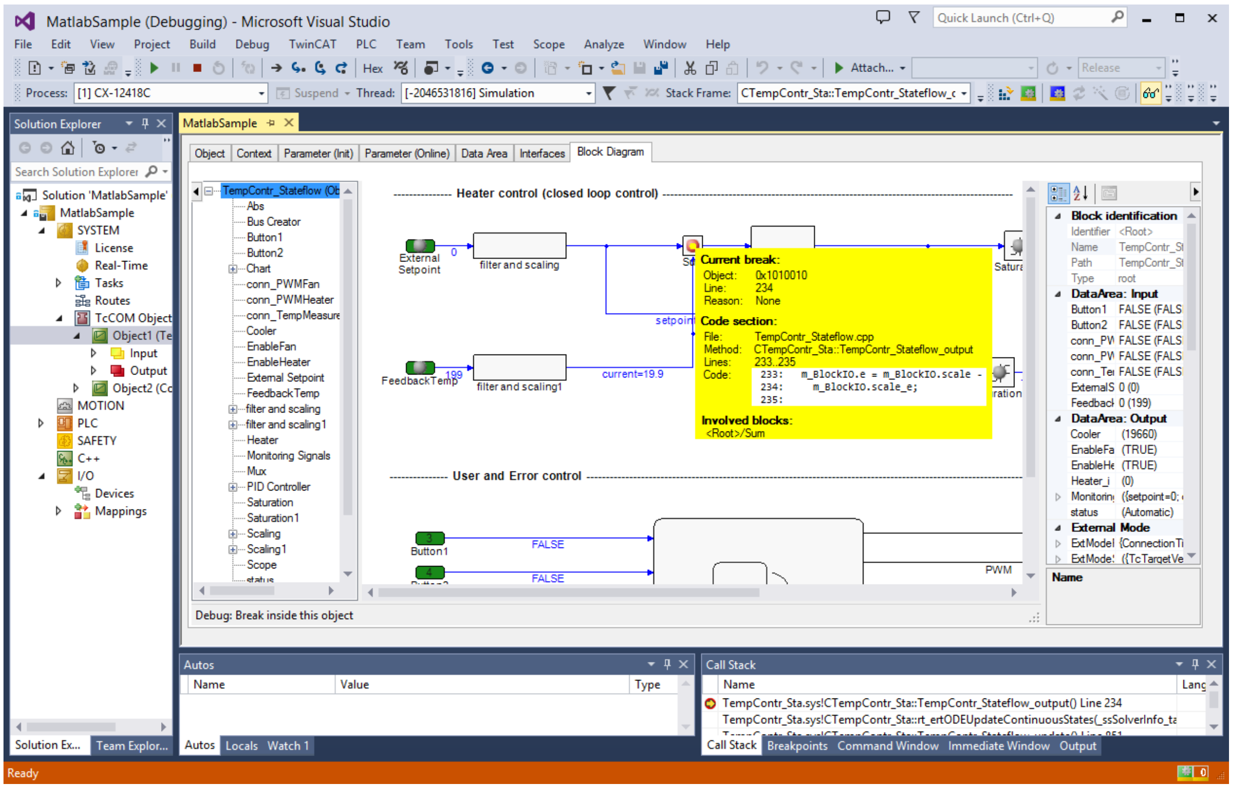
- 

Block identification	
Identifier	<Root>
Name	TempContr_Sta
Path	TempContr_Sta
Type	root
DataArea: Input	
Button1	FALSE (FALSE
Button2	FALSE (FALSE
conn_PW	FALSE (FALSE
conn_PW	FALSE (FALSE
conn_Tem	FALSE (FALSE
ExternalSe	0 (0)
Feedback	0 (199)
DataArea: Output	
Cooler	(19660)
EnableFan	(TRUE)
EnableHe	(TRUE)
Heater_i	(0)
Monitoring	({setpoint=0; c
status	(Automatic)
External Mode	
ExtModeP	{ConnectionTin
ExtModeS	({TcTargetVer
ExtModeS	({IncomingPktI

Call Stack

Name	Lang
TempContr_Sta.sys!CTempContr_Sta::TempContr_Stateflow_output() Line 234	
TempContr_Sta.sys!CTempContr_Sta::rt_ertODEUpdateContinuousStates(_ssSolverInfo_ta	

2. Additional information, such as the corresponding C++ code section, can be found in the tooltip for the breakpoint:



Current break:

Object:	0x1010010
Line:	234
Reason:	None

Code section:

File:	TempContr_Stateflow.cpp
Method:	CTempContr_Sta::TempContr_Stateflow_output
Lines:	233..235

Code:

```
233:   m_BlockIO.e = m_BlockIO.scale -
234:     m_BlockIO.scale_e;
235:
```

Involved blocks:
<Root>/Sum

Call Stack:

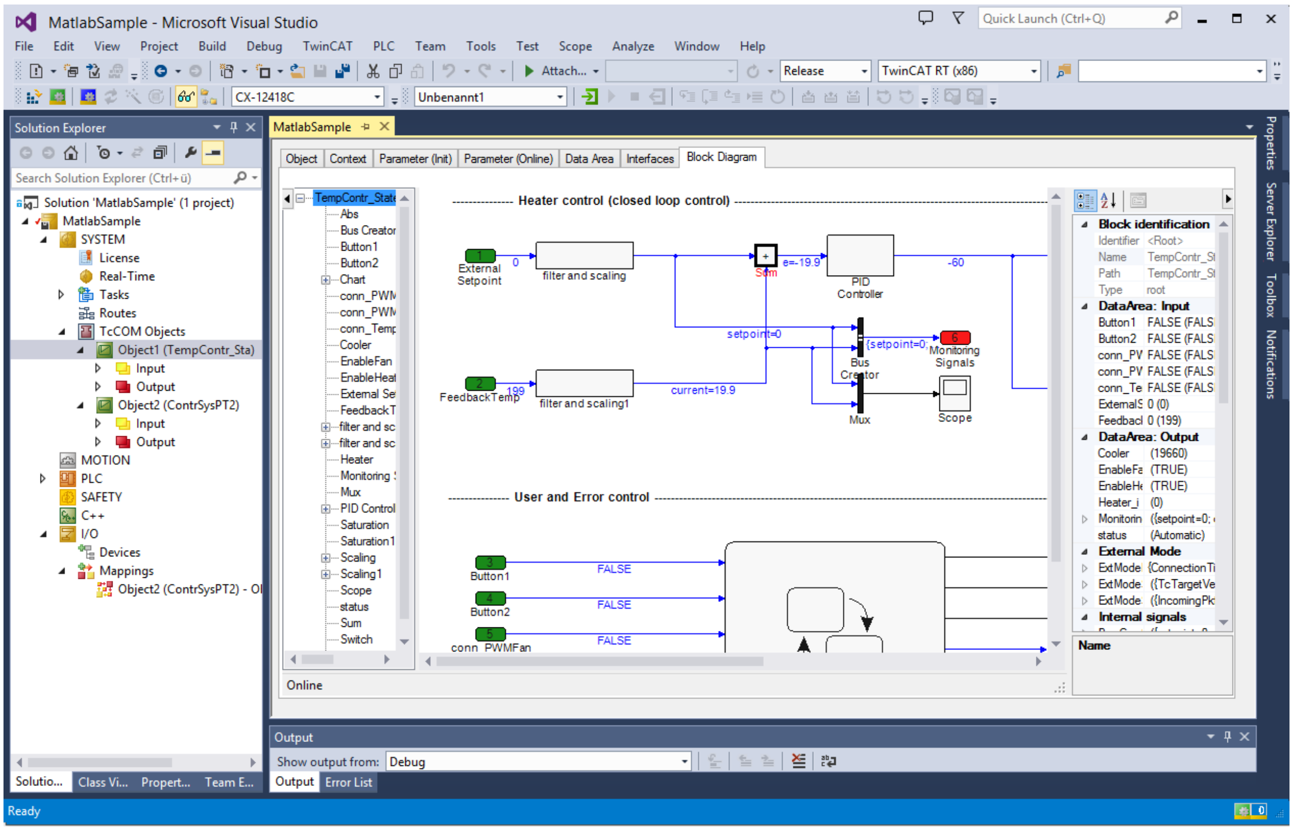
```
TempContr_Sta.sys!CTempContr_Sta::TempContr_Stateflow_output() Line 234
TempContr_Sta.sys!CTempContr_Sta::rt_ertODEUpdateContinuousStates(_ssSolverInfo_ta
```

Debug: Break inside this object

Breakpoints are not always assigned to a single function block. In many cases, the functions of several blocks are consolidated in a code section or even a line in the underlying C++ code. This means that several blocks can share the same breakpoint. Therefore, activation of a breakpoint in the block diagram may also result in changes in the point display in other blocks.

Evaluating exceptions

If exceptions occur during processing of a TcCOM module, such as division by zero, the point at which the exception occurred can be shown in the block diagram. To this end, the TcCOM module must meet the above requirements, and the C++ debugger must be enabled in the TwinCAT project (TwinCAT 3 C++ Enable C++ debugger). After the debugger has been attached, which may be done before the exception has occurred or indeed after, the block that caused the exception is highlighted in the block diagram, provided the line of code responsible for the exception can be allocated to a block. The name of the function block is shown in red, and the function block itself is marked in bold.



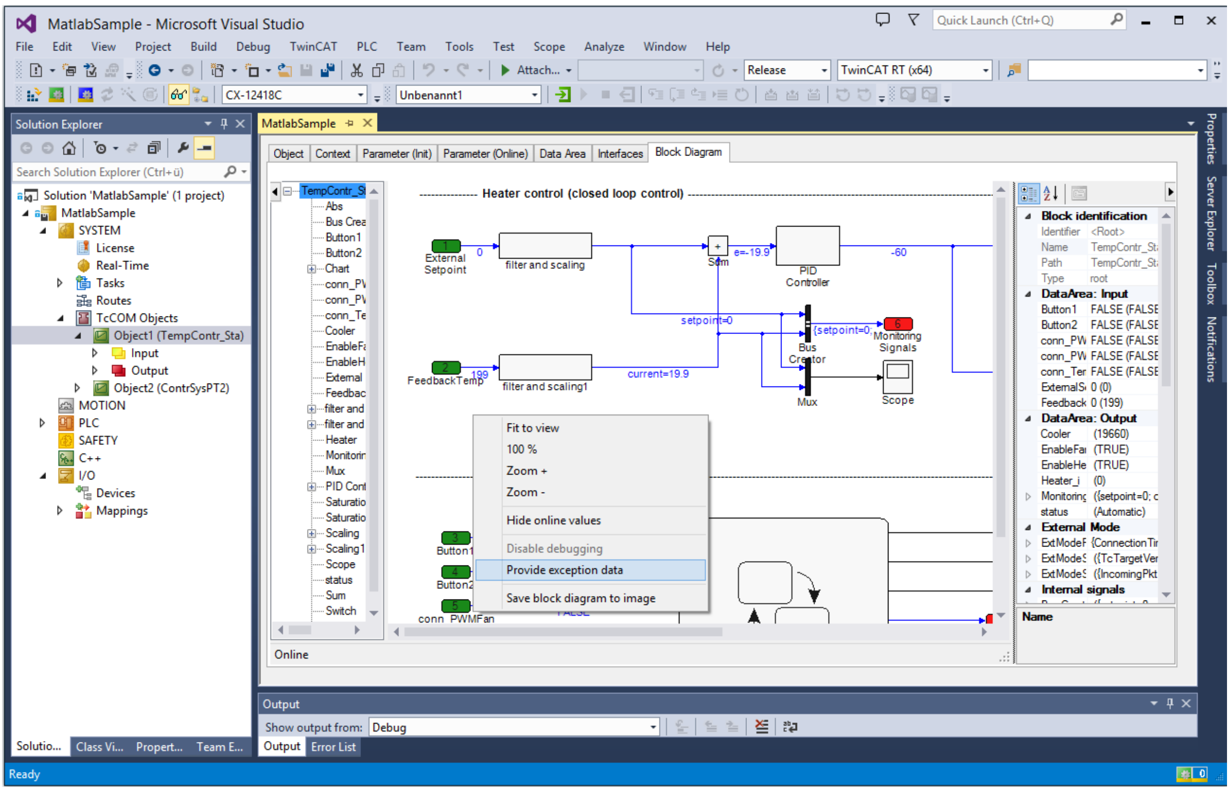
Manual evaluation of exceptions without source code

Even if the module source code is not available on the engineering system or the C++ debugger was not activated, you can highlight the error location in the block diagram once an exception has occurred.

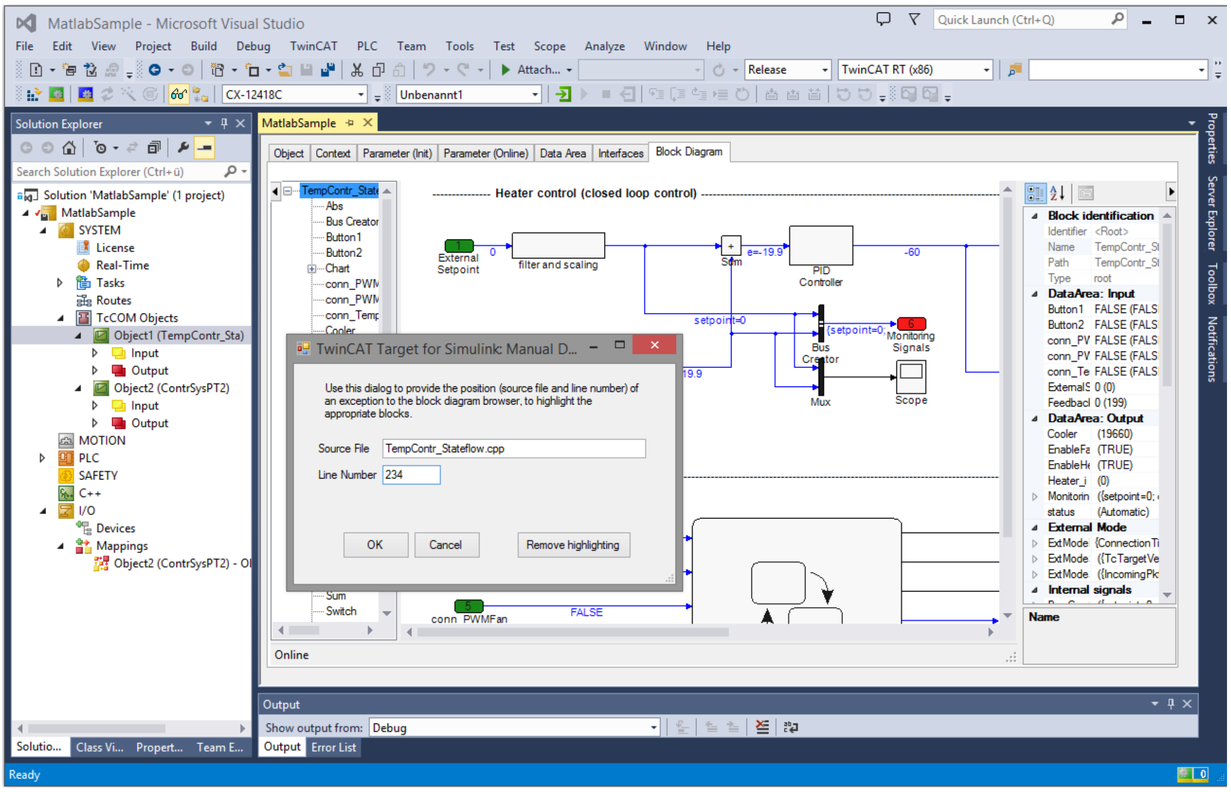
Typically, an error message will always be generated when an error occurs, indicating the source file and the line in the source code. In many cases, this information can be used to allocate an exception to a block in the block diagram. To do this, you can proceed as follows:

✓ A prerequisite for highlighting the error location within the block diagram is that debug information was generated (option **Export block diagram debug information** in the coder settings under **Tc Advanced**).

3. From the context menu of the block diagram select the entry **Provide exception data**:



4. In the dialog that opens, enter the source code file and line number provided in the error message:



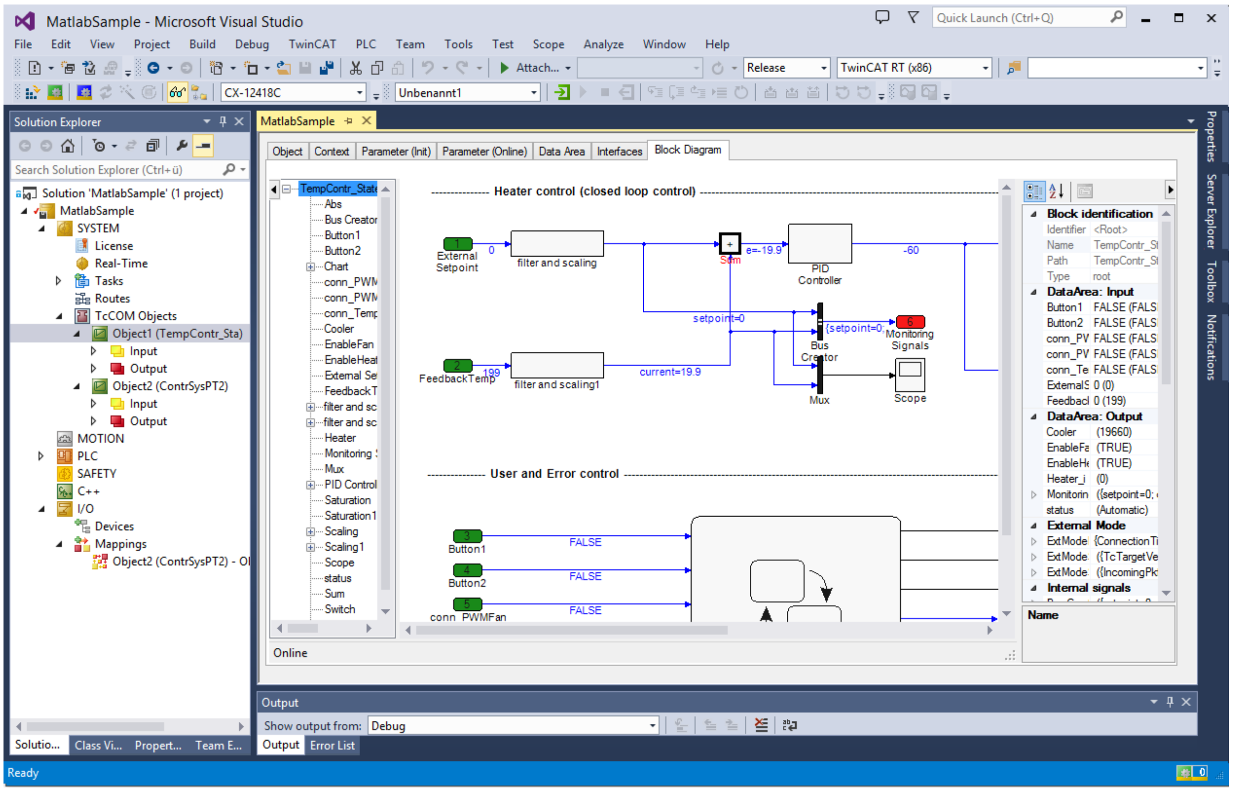
TwinCAT Target for Simulink: Manual D...

Use this dialog to position (source file and line number) of an exception to the block diagram browser, to highlight the appropriate blocks.

Source File: TempContr_Stateflow.cpp

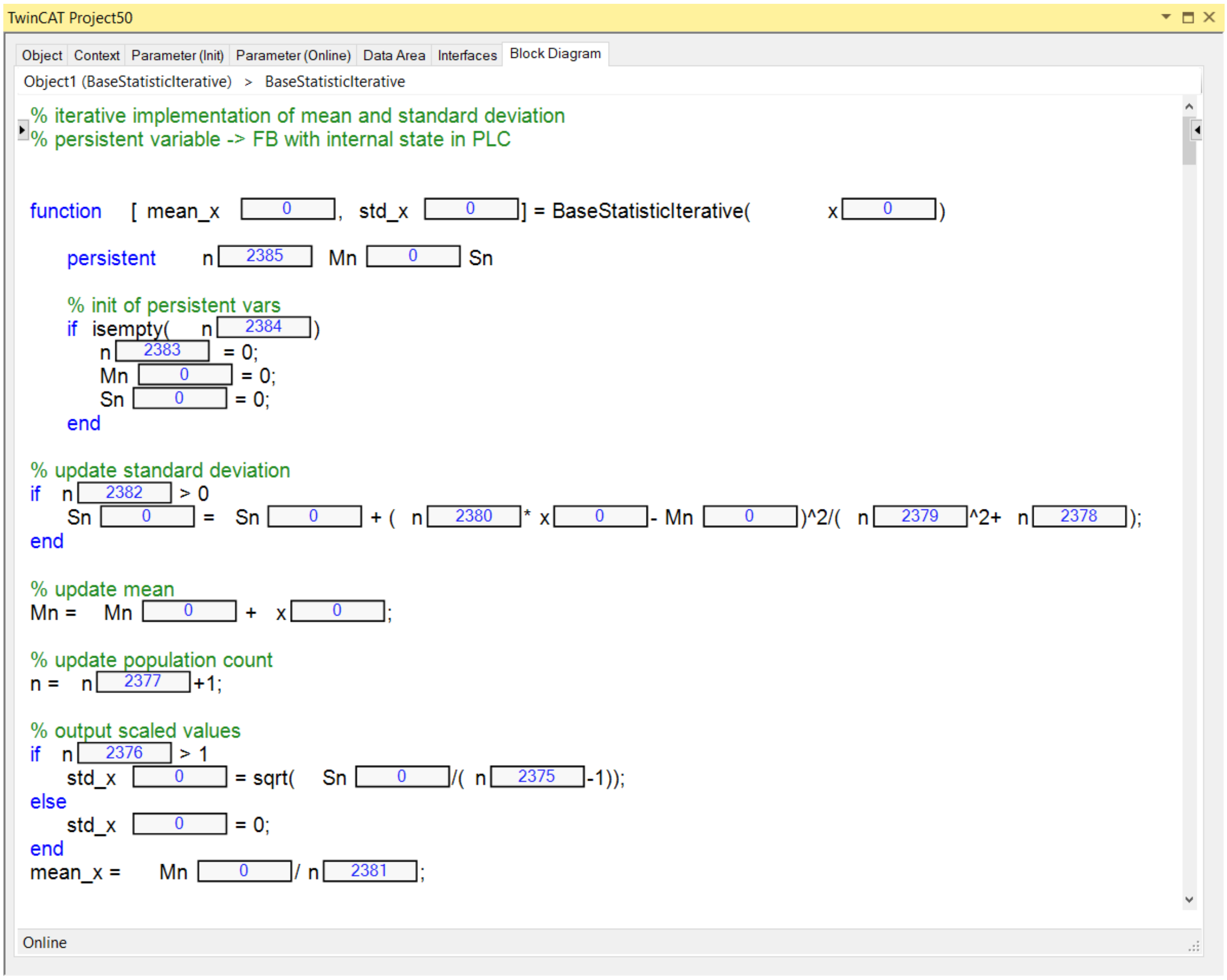
Line Number: 234

5. The name of the function block associated with the line number is displayed in red, and the function block itself is marked in bold:


MatlabSample - Microsoft Visual Studio
Solution Explorer
MatlabSample
Object Context Parameter (Init) Parameter (Online) Data Area Interfaces Block Diagram
Heater control (closed loop control)
User and Error control
Block identification
DataArea: Input
DataArea: Output
External Mode
Internal signals
Online
Output
Ready

3.2 MATLAB®-TcCOM

If a TwinCAT object was created with the TwinCAT Target for MATLAB® and the MATLAB® code export was executed, the MATLAB® code of the MATLAB® function can be displayed as a control in TwinCAT XAE.

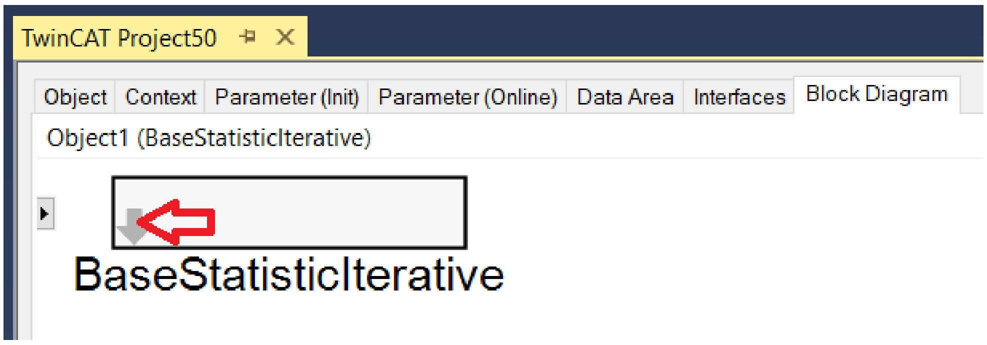


```
% iterative implementation of mean and standard deviation
% persistent variable -> FB with internal state in PLC

function   [ mean_x  0 ,  std_x  0 ] = BaseStatisticIterative(        x 0 )

      persistent      n 2385  Mn  0  Sn

      % init of persistent vars
      if  isempty(    n 2384 )
           n 2383  = 0;
           Mn  0  = 0;
           Sn  0  = 0;
      end

% update standard deviation
if   n 2382  > 0
      Sn  0  =  Sn  0  + (  n 2380 * x 0 - Mn  0 )^2/(  n 2379 ^2+  n 2378 );
end

% update mean
Mn =   Mn  0  +  x 0 ;

% update population count
n =   n 2377 +1;

% output scaled values
if   n 2376  > 1
      std_x  0  = sqrt(    Sn  0 /(  n 2375 -1));
else
      std_x  0  = 0;
end
mean_x =     Mn  0 / n 2381 ;
```

3.2.1 Operation of the block diagram window

The export of the MATLAB® code can be configured during generation of a TcCOM module from MATLAB®. If the export was enabled, the code can be found in the TwinCAT development environment under the **Block Diagram** tab of the module instance.

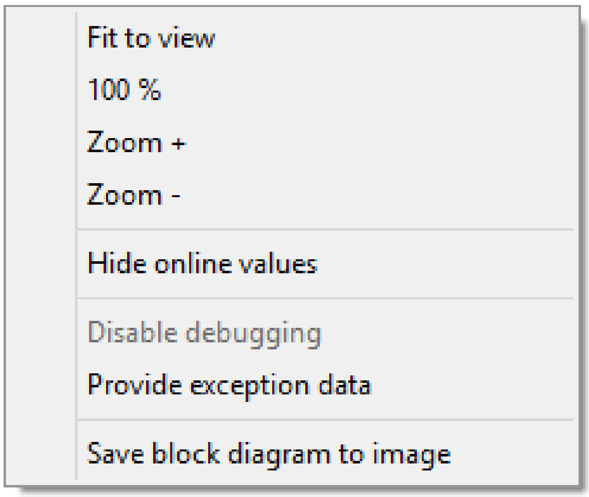
On the top level you will find the created module in block representation. Select the gray arrow in the block to display the content.



Shortcut functions:

Shortcut	Function
Space	Zoom to current size of the block diagram tab
Backspace	Switch to the next higher hierarchical level
ESC	Switch to the next higher hierarchical level
CTRL + "+"	Zoom in
CTRL + "-"	Zoom out
F5	Attach Debugger (*System- > Real-Time -> C++ Debugger -> Enable C++ Debugger* must be activated)

Context menu functions:

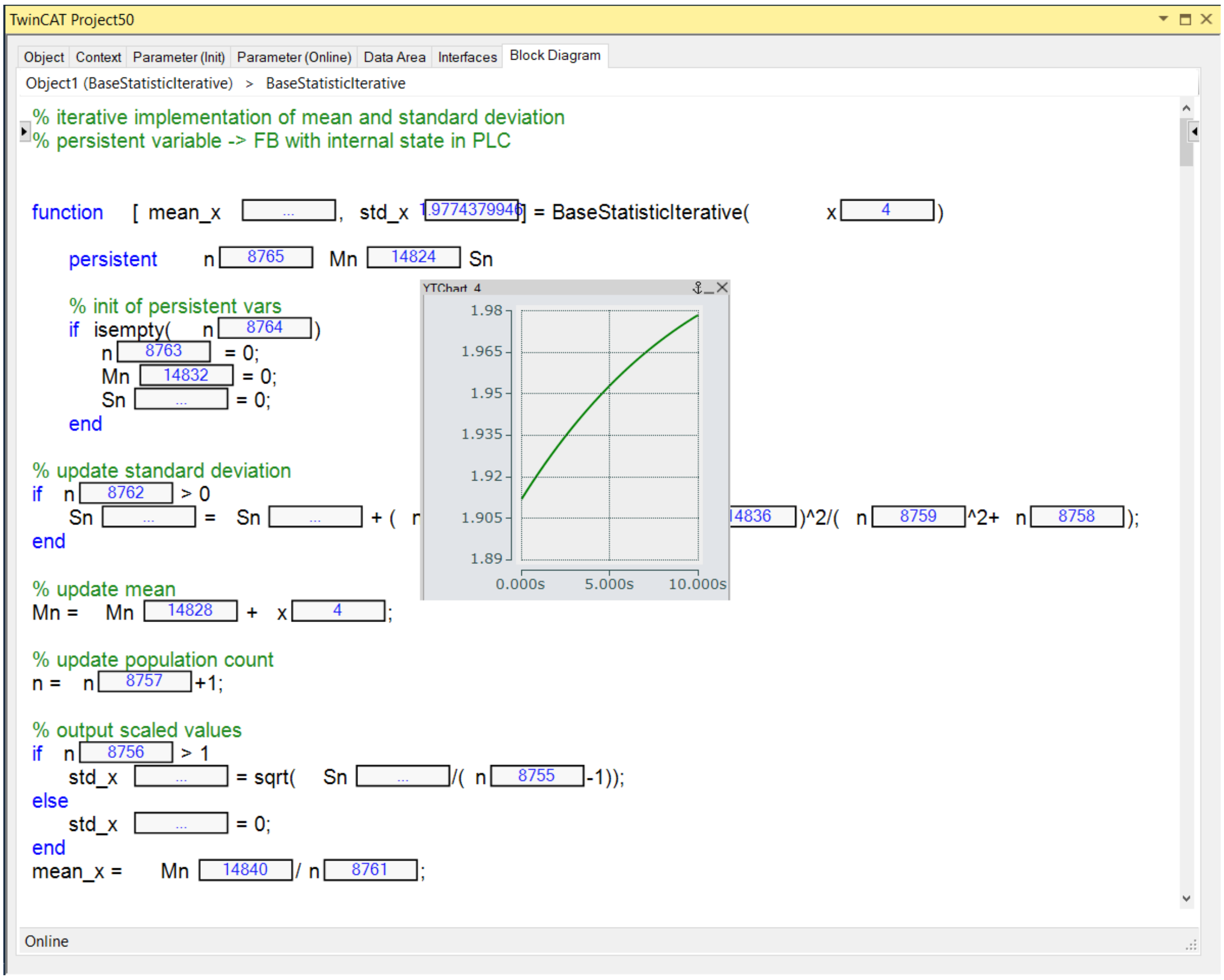


3.2.2 Display of signal curves

Selected variables can be retrieved in TwinCAT XAE via ADS. It is therefore possible to display them in a mini-scope within the block diagram, or with the TwinCAT Scope within a measurement project.

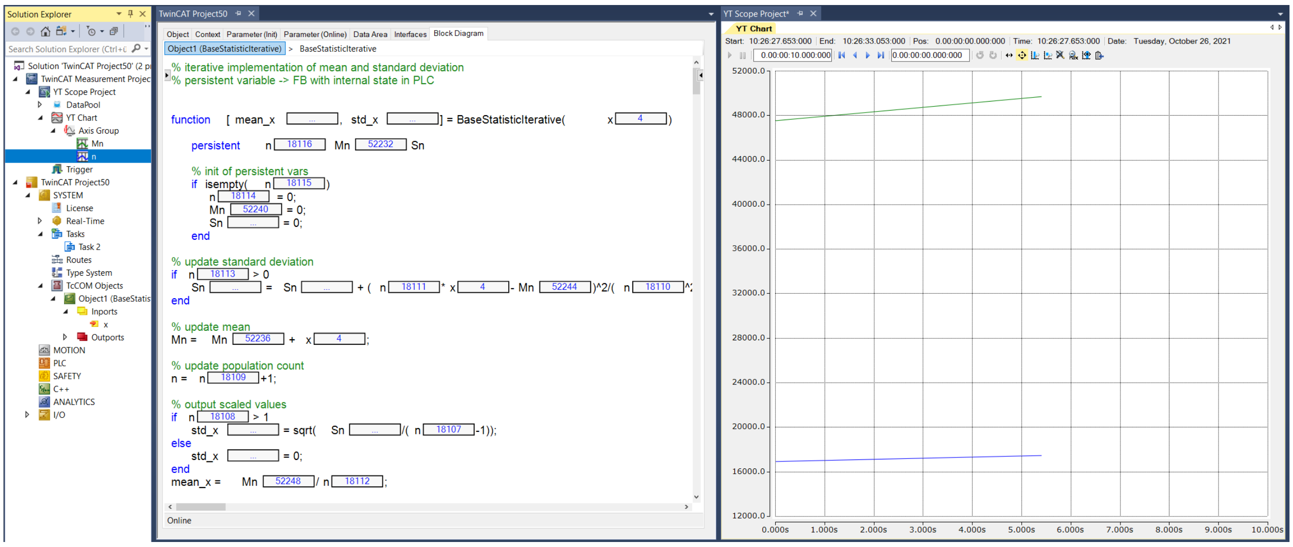
Variables that can be displayed in scope have a trailing black frame in the code display. In this frame, the values are displayed in blue during operation.

Drag&Drop a "blue variable" onto the block diagram window to open a Mini.Scope.



```
% iterative implementation of mean and standard deviation
% persistent variable -> FB with internal state in PLC


function   [ mean_x [ ... ],  std_x [1.9774379946] = BaseStatisticIterative(        x[ 4 ])

     persistent      n[ 8765 ]  Mn[ 14824 ] Sn

     % init of persistent vars
     if  isempty(   n[ 8764 ])
          n[ 8763 ] = 0;
          Mn[ 14832 ] = 0;
          Sn[ ... ] = 0;
     end

% update standard deviation
if   n[ 8762 ] > 0
     Sn[ ... ]  =   Sn[ ... ] + (  r                    [14836 ])^2/(   n[ 8759 ]^2+   n[ 8758 ]);
end

% update mean
Mn =    Mn[ 14828 ]  +   x[ 4 ];

% update population count
n =    n[ 8757 ]+1;

% output scaled values
if   n[ 8756 ] > 1
     std_x[ ... ] = sqrt(   Sn[ ... ]/(  n[ 8755 ]-1));
else
     std_x[ ... ] = 0;
end
mean_x =      Mn[ 14840 ]/ n[ 8761 ];
```

By dragging and dropping a "blue variable" onto the Axis Group of a chart in the TwinCAT Measurement project, the variables are added to the TwinCAT Scope.



```
% iterative implementation of mean and standard deviation
% persistent variable -> FB with internal state in PLC


function   [ mean_x [ ... ],  std_x [ ... ]] = BaseStatisticIterative(        x[ 4 ])

     persistent      n[ 18116 ]  Mn[ 52232 ] Sn

     % init of persistent vars
     if  isempty(   n[ 18115 ])
          n[ 18114 ] = 0;
          Mn[ 52240 ] = 0;
          Sn[ ... ] = 0;
     end

% update standard deviation
if   n[ 18113 ] > 0
     Sn[ ... ]  =   Sn[ ... ] + (  n[ 18111 ]* x[ 4 ]- Mn[ 52244 ])^2/(   n[ 18110 ]^
end

% update mean
Mn =    Mn[ 52236 ]  +   x[ 4 ];

% update population count
n =    n[ 18109 ]+1;

% output scaled values
if   n[ 18108 ] > 1
     std_x[ ... ] = sqrt(   Sn[ ... ]/(  n[ 18107 ]-1));
else
     std_x[ ... ] = 0;
end
mean_x =      Mn[ 52248 ]/ n[ 18112 ];
```

Which variables are visible as "blue variables" in TwinCAT XAE?

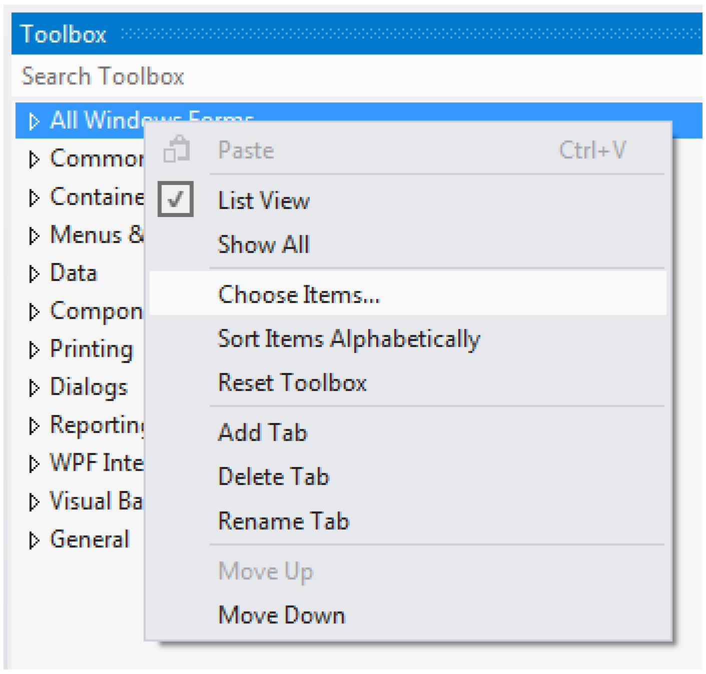
- The input variables
- The output variables
- Persistent variables (MATLAB definition `persistent var1 … varN`)

3.3 Integrating the block diagram controls

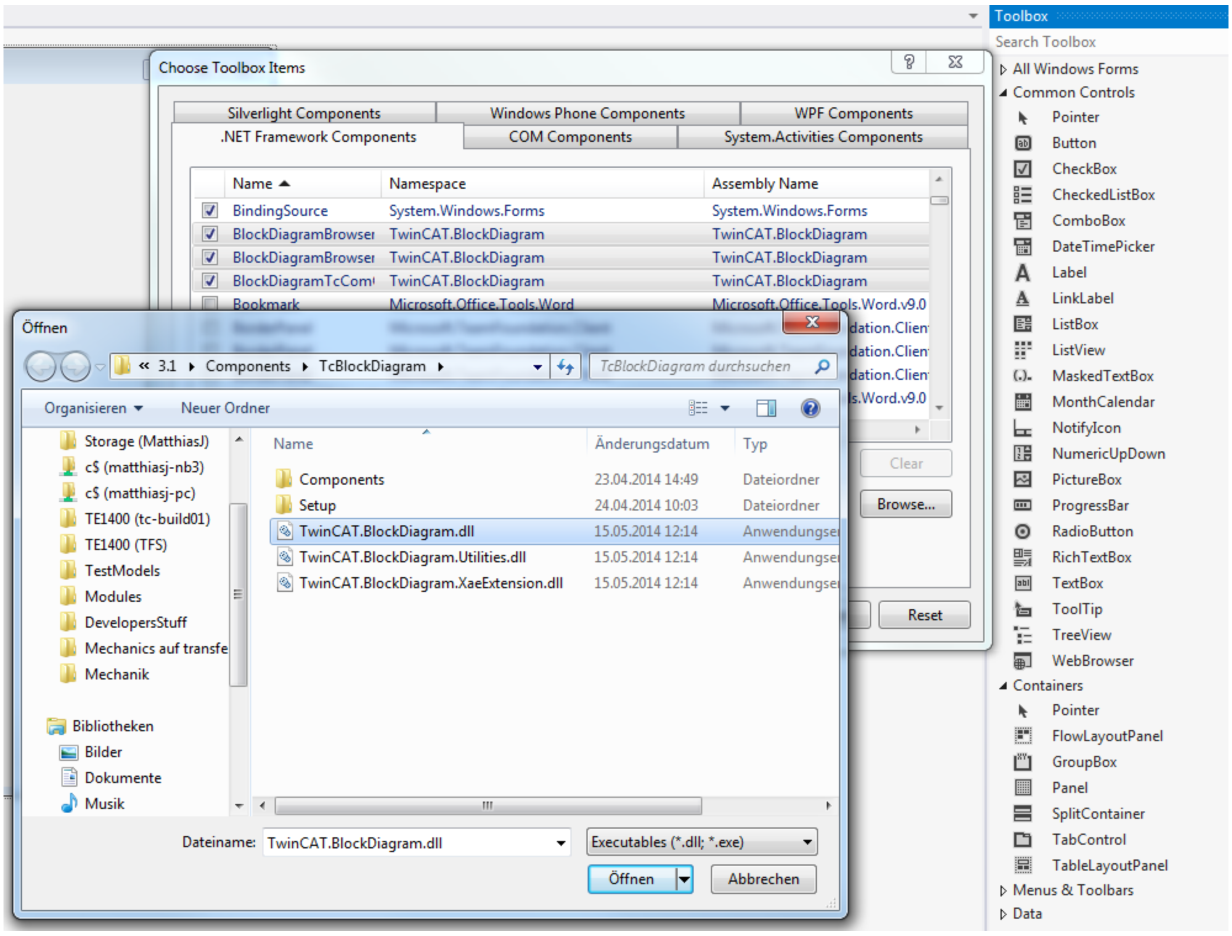
The control that displays the block diagram in the TwinCAT XAE environment can also be integrated as a control in your own visualizations.

The following steps are required:

1. Create a new Windows Forms application.
2. Add TwinCAT.BlockDiagram.dll to the toolbox:
3. To do this, select the "Choose Items..." entry in the context menu.



4. Navigate to TwinCAT.Blockdiagram.dll, which can be found under <*TwinCAT installation path*>*\3.1\Components\TcBlockDiagram*.

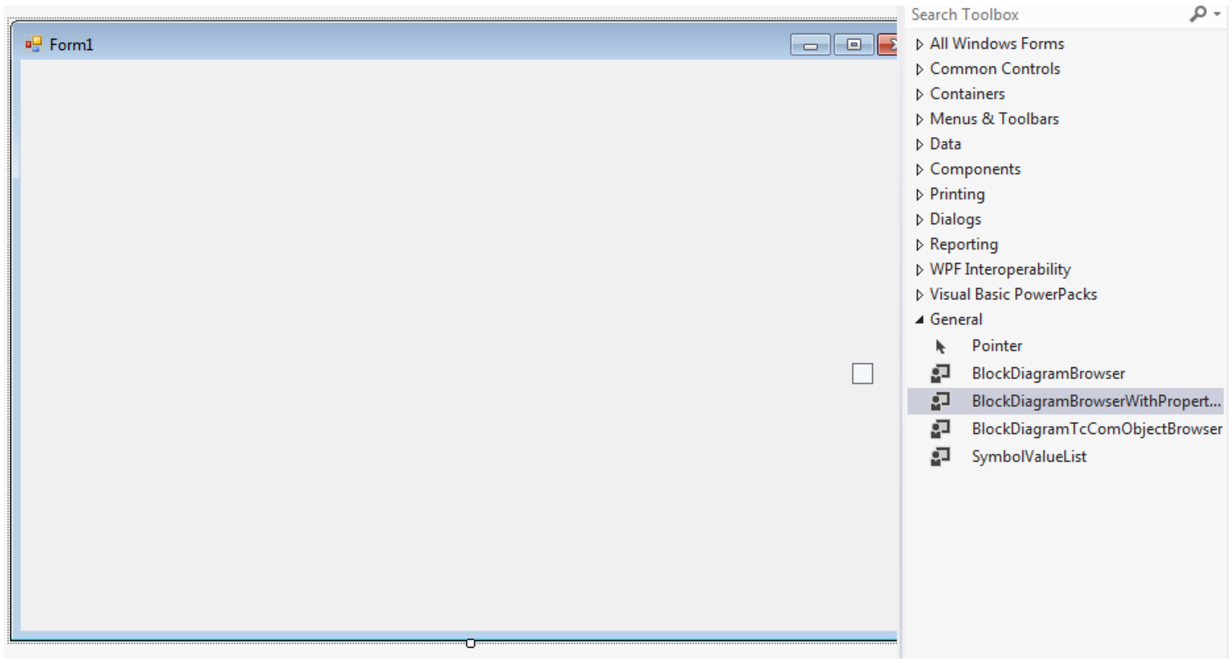


	Name	Namespace	Assembly Name
☑	BindingSource	System.Windows.Forms	System.Windows.Forms
☑	BlockDiagramBrowser	TwinCAT.BlockDiagram	TwinCAT.BlockDiagram
☑	BlockDiagramBrowser	TwinCAT.BlockDiagram	TwinCAT.BlockDiagram
☑	BlockDiagramTcCom	TwinCAT.BlockDiagram	TwinCAT.BlockDiagram
☐	Bookmark	Microsoft.Office.Tools.Word	Microsoft.Office.Tools.Word.v9.0

Name	Änderungsdatum	Typ
Components	23.04.2014 14:49	Dateiordner
Setup	24.04.2014 10:03	Dateiordner
TwinCAT.BlockDiagram.dll	15.05.2014 12:14	Anwendungse
TwinCAT.BlockDiagram.Utilities.dll	15.05.2014 12:14	Anwendungse
TwinCAT.BlockDiagram.XaeExtension.dll	15.05.2014 12:14	Anwendungse

Dateiname: TwinCAT.BlockDiagram.dll
Executables (*.dll; *.exe)

5. Add a TcBlockdiagram control instance to the Windows Forms object using drag and drop.



4 TwinCAT Automation Interface: use in MATLAB®

Short description of the Automation Interface

TwinCAT XAE configurations can be automatically generated and edited via programming/script codes using the TwinCAT Automation Interface. The automation of a TwinCAT configuration is available thanks to so-called Automation Interfaces, which can be accessed via all COM-capable programming languages (e.g. C++ or .NET) and also via dynamic script languages such as Windows PowerShell, IronPython or even the (obsolete) Vbscript. Use from the MATLAB® environment is also possible.

Detailed documentation of the product can be found here: TwinCAT Automation Interface

Use in MATLAB®

The Automation Interface can be made visible in MATLAB® through the command NET.addAssembly. This will enable you to use the interfaces (Automation Interface API) described in the product documentation. You can also find many programming samples for use from C# and PowerShell (Automation Interface Configuration).

In order to simplify the entry from MATLAB® for you, you can find below a sample implementation for MATLAB® on the basis of a MATLAB® class, which you can use, modify and expand.

4.1 Sample: Tc3AutomationInterface

Overview

The sample code consists of two m-files:

- *Tc3AutomationInterface.m*: MATLAB® class that implements several frequently used methods.
- *Tc3AutomationInterfaceGuide.mlx*: MATLAB live script that calls the MATLAB® class as an example.

Call sample with MATLAB®

The TwinCAT Tool for MATLAB® and Simulink® Setup installs the sample on your system. Call the sample with the MATLAB® Command Window:
`TwinCAT.ModuleGenerator.Samples.Start('AutomationInterface').`

The MATLAB® script

The MATLAB® script provides a sample of how you can generate a TwinCAT solution, scan the EtherCAT master for I/Os, instantiate two TcCOM modules, link them and activate the project on a target.

In order to be able to run the script, the two TcCOMs used must be present in your *publish directory %TwinCATDir%\\CustomConfig\Modules\*. For this, download the Temperature Controller sample from the TE1400 | Target for MATLAB®/Simulink®. Then copy the file folder from the directory . *\TE1400Sample_TemperatureController_PrecompiledTcComModules\Actual TwinCAT versions* into the *publish directory*.

Run the m-file *Tc3AutomationInterface_Testbench.m*. The latest Visual Studio instance available on your system is opened in the background and the TwinCAT solution is configured, saved and activated.

The MATLAB® class

The properties

All variables and interfaces belonging to the instance of the class are contained in the properties of the *Tc3AutomationInterface* class. Hence, several TwinCAT solutions can be built up in a MATLAB® script by generating an instance of the class for each solution. There are then no overlaps.

The constructor

```
function this = Tc3AutomationInterface
```

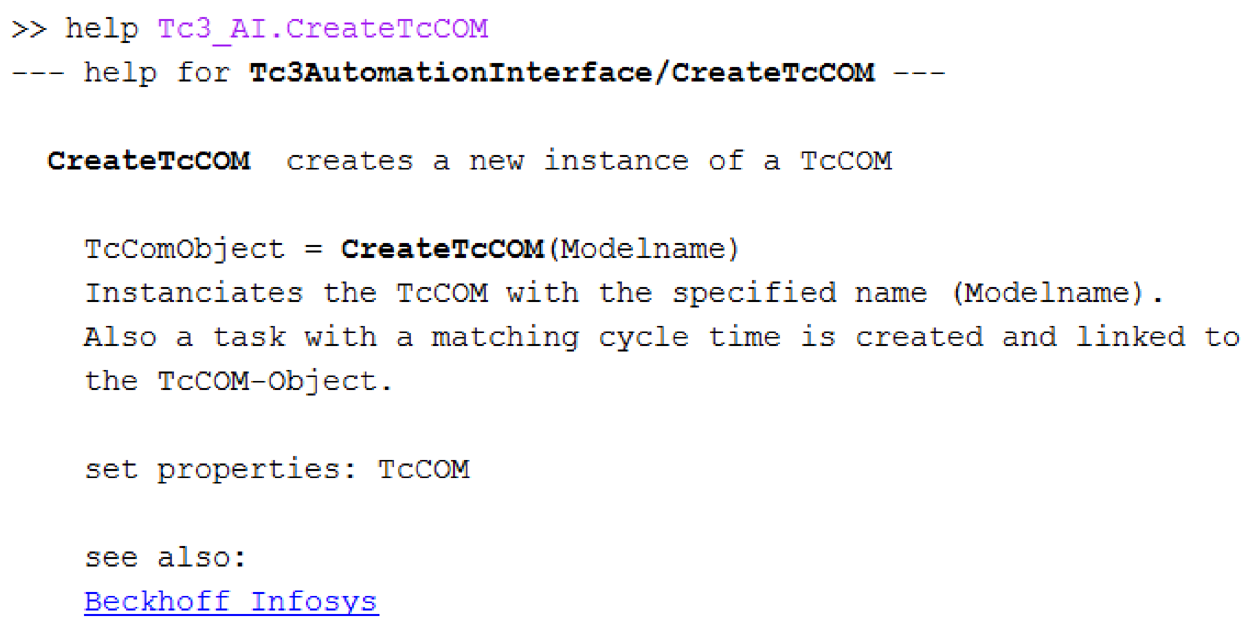
The constructor loads all necessary assemblies and, if successful, sets the AssembliesLoaded property to TRUE. The loaded assemblies are:

- EnvDTE and EnvDTE80: libraries for the Visual Studio Core Automation. Necessary for the configuration of Visual Studio.
- TCatSysManagerLib: TwinCAT Automation Interface library for the configuration of a TwinCAT solution in Visual Studio.
- TwinCAT.Ads: ADS library, e.g. for reading and changing the XAR state.
- System.Xml: library for parsing XML files.

Selected methods of the class

```
function TcComObject = CreateTcCOM(this, Modelname)
```

Use the MATLAB® help functions in order to view the function and the parameters of the method.



```
>> help Tc3_AI.CreateTcCOM
--- help for Tc3AutomationInterface/CreateTcCOM ---

  CreateTcCOM  creates a new instance of a TcCOM

    TcComObject = CreateTcCOM(Modelname)
    Instanciates the TcCOM with the specified name (Modelname).
    Also a task with a matching cycle time is created and linked to
    the TcCOM-Object.

    set properties: TcCOM

    see also:
    Beckhoff Infosys
```

A link to the Beckhoff Infosys is also offered with some methods. These refer to documentation examples from the TwinCAT Automation Interface documentation, so that you can directly view a comparison of the implementation in MATLAB®, C# and PowerShell. You can also find a link to the Beckhoff Infosys in the comment in some sections, allowing you to view the source of the information.

The CreateTcCOM method initially begins with the parsing of the *<modelname>.tmc* file, from which the ClassID, the task cycle time and the task priority are extracted with *System.Xml*. A corresponding TcCOM is then instantiated and one (or more) associated tasks generated with the Automation Interface. Finally, the task is/tasks are assigned to the TcCOM.

```
function ActivateOnDevice(this, AmsNetId)
```

TwinCAT ADS is used in order to query or change the current status of a TwinCAT runtime, e.g. config or run. In the ActivateOnDevice method the XAR is initially switched to the config mode with the specified AmsNetId and the current TwinCAT configuration is then activated and the system started. Pauses are entered between the individual steps, as this procedure may need a little time.

Static methods

Static methods are also available even without an instance of the class.

```
function vsVersions = GetInstalledVisualStudios
```

A function that detects and lists the Visual Studio installations available on the system via the Register Key entries is prepared here. The implementation is limited to VS 2010 to VS 2017.

Documents about this

https://infosys.beckhoff.com/content/1033/tc3_matlab_overview/Resources/5776206091.zip

More Information:
www.beckhoff.com/te1000

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