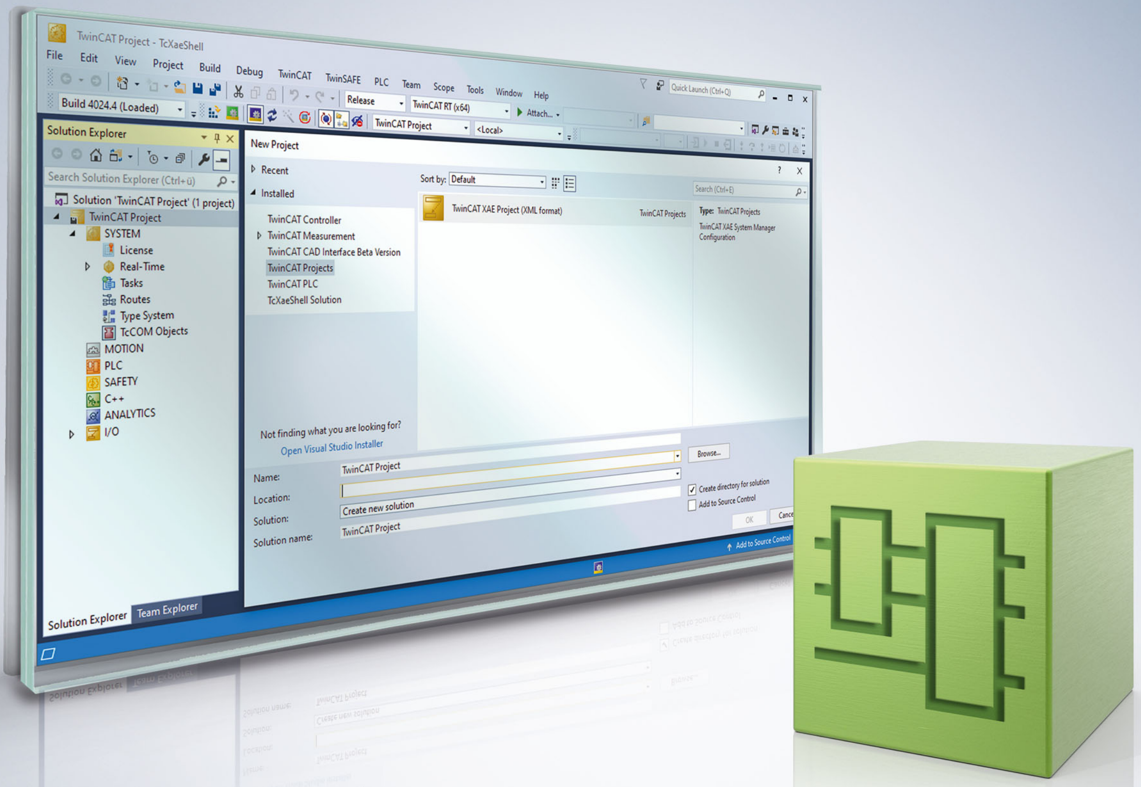
BECKHOFF New Automation Technology

Manual | EN

TE1000

TwinCAT 3 | PLC Lib: Tc2_SystemCX



2023-12-11 | Version: 1.6.1

Inhaltsverzeichnis

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 here.

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.

2 Introduction

This library contains functions and function blocks that use features on the devices of the Embedded PC CX series.

Function blocks

Name	Description
FB_CXProfiler [▸ 13]	Runtime measurement of PLC code via the CPU counter
FB_CXSetTextDisplay [▸ 16]	Control of the two-line display of the CX1100
FB_CXSetTextDisplayUSB [▸ 17]	Writing and deleting of rows on the two-line display of the CX 2100 or the EL6090 terminal.
FB_CXGetTextDisplayUSB [▸ 20]	Reading of rows on the two-line display of the CX 2100 or the EL6090 terminal.
FB_CXSimpleUps [▸ 21]	Control of the UPS CX1190-UPS (device name CX1100-0900, CX1100-0910, CX1100-0920)
FB_CX5010SetWatchdog [▸ 23]	Activates a hardware watchdog on the CX5010.
FB_CX5020SetWatchdog [▸ 24]	Activates a hardware watchdog on the CX5020.

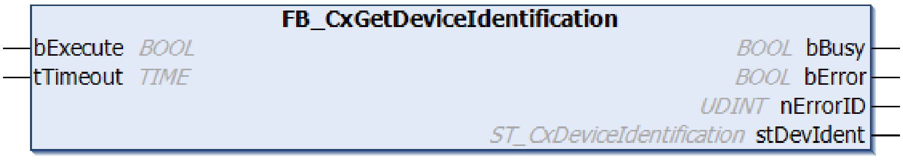
Functions

Name	Description
F_CXSubTimeStamp [▸ 35]	Calculates 64-bit subtraction (time A [100 ns] - time B [100 ns]) as result in µs; only for differences between 0 and 4294967295 us; see link.
F_CXNaviSwitch [▸ 37]	This function converts the value of the CX1100 navigation switch to an enum value.
F_CXNaviSwitchUSB [▸ 38]	This function converts the value of the CX2100 navigation switch to an enum value.

3 Function Blocks

3.1 [obsolete]

3.1.1 FB_CxGetDeviceIdentification


FB_CxGetDeviceIdentification
bExecute BOOL
tTimeout TIME
BOOL bBusy
BOOL bError
UDINT nErrorID
ST_CxDeviceIdentification stDevIdent

The function block FB_CxGetDeviceIdentification can be used to read device ID data of the CX.

Obsolete functionality

- Use the FB_GetDeviceIdentificationEx from the Tc2_Utilities library.

Inputs

```
VAR_INPUT
    bExecute : BOOL;
    tTimeout : TIME;
END_VAR
```

Name	Type	Description
bExecute	BOOL	The command is executed with the rising edge.
tTimeout	TIME	States the time before the function is canceled.

Outputs

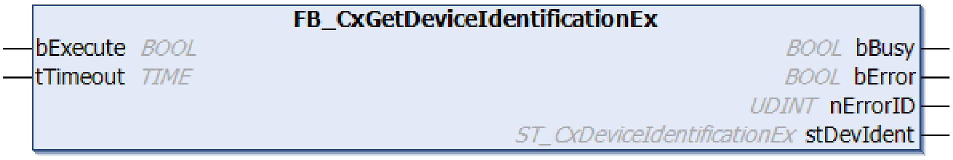
```
VAR_OUTPUT
    bBusy      : BOOL;
    bError     : BOOL;
    nErrorID   : UDINT;
    stDevIdent : ST_CxDeviceIdentification;
END_VAR
```

Name	Type	Description
bBusy	BOOL	The data are read from the CX. After error-free execution, the data are in the structure stDevIdent if bBusy = FALSE.
bError	BOOL	Becomes TRUE, as soon as an error occurs.
nErrorID	UDINT	Supplies the error number when the bError output is set.
stDevIdent	ST_CxDeviceIdentification	Contains the read device data. (Type: ST_CxDeviceIdentification [▶ 42])

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1	CX (WES7/Win7/Win10: TC RT x86/x64, WEC6/7 :TC TR x86, WEC7: TC CE7 ARMV7)	Tc2_SystemCX

3.1.2 FB_CxGetDeviceIdentificationEx


FB_CxGetDeviceIdentificationEx
bExecute BOOL
tTimeout TIME
BOOL bBusy
BOOL bError
UDINT nErrorID
ST_CxDeviceIdentificationEx stDevIdent

The function block FB_CxGetDeviceIdentificationEx can be used to read device ID data of the CX. The function block is an extension of the function block FB_CxGetDeviceIdentification. The read device data are stored in the variable stDevIdent of type ST_CxDeviceIdentificationEx [▶ 42].

Obsolete functionality

- Use the FB_GetDeviceIdentificationEx from the Tc2_Utilities library.

Inputs

```
VAR_INPUT
    bExecute  :  BOOL;
    tTimeout  :  TIME;
END_VAR
```

Name	Type	Description
bExecute	BOOL	The command is executed with a rising edge.
tTimeout	TIME	States the time before the function is canceled.

Outputs

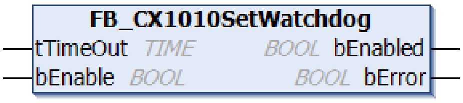
```
VAR_OUTPUT
    bBusy       :  BOOL;
    bError      :  BOOL;
    nErrorId    :  UDINT;
    stDevIdent  :  ST_CxDeviceIdentificationEx;
END_VAR
```

Name	Type	Description
bBusy	BOOL	The data are read from the CX. After error-free execution, the data are in the structure stDevIdent if bBusy = FALSE.
bError	BOOL	Becomes TRUE, as soon as an error occurs.
nErrorID	UDINT	Supplies the error number when the bError output is set.
stDevIdent	ST_CxDeviceIdentificationEx	Contains the read device data (type: ST_CxDeviceIdentificationEx [▶ 42])

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1	CX (WES7/Win7/Win10: TC RT x86/x64, WEC6/7 :TC TR x86, WEC7: TC CE7 ARMV7)	Tc2_SystemCX

3.1.3 FB_CX1010SetWatchdog



The function block FB_CX1010SetWatchdog activates a hardware watchdog on the CX1010. The watchdog is activated via bEnable = TRUE and the tTimeOut time. The tTimeOut time can be a minimum of 2 seconds and a maximum of 255 seconds.

Once the watchdog has been activated, the function block instance must be called cyclically at shorter intervals than tTimeOut, since the CX1010 restarts automatically when tTimeOut has elapsed. The watchdog can therefore be used to automatically reboot systems, which have entered an infinite loop or where the PLC has become stuck.

The watchdog can be deactivated via bEnable = FALSE or tTimeOut = T#0s.

NOTICE

The watchdog must be deactivated before breakpoints are used, before a PLC reset or an overall reset, before a TwinCAT stop, before switching to Config mode or before the configuration is activated, because otherwise the CX1010 would reboot immediately once tTimeOut has elapsed.

Inputs

```
VAR_INPUT
    tTimeOut : TIME;
    bEnable  : BOOL;
END_VAR
```

Name	Type	Description
tTimeOut	TIME	Watchdog time, after which a restart is performed.
bEnable	BOOL	Activating/deactivating the watchdog.

Outputs

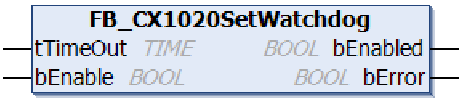
```
VAR_OUTPUT
    bEnabled : BOOL;
    bError   : BOOL;
END_VAR
```

Name	Type	Description
bEnabled	BOOL	TRUE = Watchdog is active, FALSE = Watchdog is not active.
bError	BOOL	Error when activating or deactivating the watchdog.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.0	CX (x86)	Tc2_SystemCX

3.1.4 FB_CX1020SetWatchdog



The function block FB_CX1020SetWatchdog activates a hardware watchdog on the CX1020. The watchdog is activated via bEnable = TRUE and the tTimeOut time. The tTimeOut time can be a minimum of 2 seconds and a maximum of 255 seconds.

Once the watchdog has been activated, the function block instance must be called cyclically at shorter intervals than tTimeOut, since the CX1020 restarts automatically when tTimeOut has elapsed. The watchdog can therefore be used to automatically reboot systems, which have entered an infinite loop or where the PLC has become stuck.

The watchdog can be deactivated via bEnable = FALSE or tTimeOut = T#0s.

NOTICE

The watchdog must be deactivated before breakpoints are used, before a PLC reset or an overall reset, before a TwinCAT stop, before switching to Config mode or before the configuration is activated, because otherwise the CX1020 would reboot immediately once tTimeOut has elapsed.

Inputs

```
VAR_INPUT
    tTimeOut : TIME;
    bEnable  : BOOL;
END_VAR
```

Name	Type	Description
tTimeOut	TIME	Watchdog time, after which a restart is performed.
bEnable	BOOL	Activating/deactivating the watchdog.

Outputs

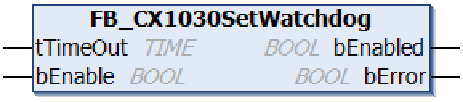
```
VAR_OUTPUT
    bEnabled : BOOL;
    bError   : BOOL;
END_VAR
```

Name	Type	Description
bEnabled	BOOL	TRUE = Watchdog is active, FALSE = Watchdog is not active.
bError	BOOL	Error when activating or deactivating the watchdog.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.0	CX (x86)	Tc2_SystemCX

3.1.5 FB_CX1030SetWatchdog



The function block FB_CX1030SetWatchdog activates a hardware watchdog on the CX1030. The watchdog is activated via bEnable = TRUE and the tTimeOut time. The tTimeOut time can be a minimum of 2 seconds and a maximum of 255 seconds.

Once the watchdog has been activated, the function block instance must be called cyclically at shorter intervals than tTimeOut, since the CX1030 restarts automatically when tTimeOut has elapsed. The watchdog can therefore be used to automatically reboot systems, which have entered an infinite loop or where the PLC has become stuck.

The watchdog can be deactivated via bEnable = FALSE or tTimeOut = T#0s.

NOTICE

The watchdog must be deactivated before breakpoints are used, before a PLC reset or an overall reset, before a TwinCAT stop, before switching to Config mode or before the configuration is activated, because otherwise the CX1030 would reboot immediately once tTimeOut has elapsed.

Inputs

```
VAR_INPUT
    tTimeOut : TIME;
    bEnable  : BOOL;
END_VAR
```

Name	Type	Description
tTimeOut	TIME	Watchdog time, after which a restart is performed.
bEnable	BOOL	Activating/deactivating the watchdog.

Outputs

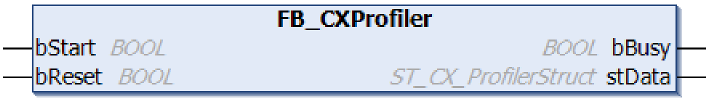
```
VAR_OUTPUT
    bEnabled : BOOL;
    bError   : BOOL;
END_VAR
```

Name	Type	Description
bEnabled	BOOL	TRUE = Watchdog is active, FALSE = Watchdog is not active.
bError	BOOL	Error when activating or deactivating the watchdog.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.0	CX (x86)	Tc2_SystemCX

3.2 FB_CXProfiler



The function block FB_CXProfiler can be used to measure the execution time of the PLC code under Microsoft Windows CE.

For other operating systems, refer to the Profiler function block in the Tc2_Utilities library.

Internally, an instance of the GETCPUCOUNTER function block is called. The measurement is started by a rising edge at the bStart input, and is stopped by a falling edge. The measurements are evaluated internally, and are then made available for further processing at the stData output in a structure of type ST_CX_ProfilerStruct [▶ 43]. As well as the current, minimum and maximum execution times, the function block calculates the mean execution time for the last 100 measurements. The times measured are given in microseconds. The output variable stData.dwMeasureCycle [▶ 43] provides information about the number of measurements that have already been carried out. In order to measure the execution time for a specific segment of the PLC program the measurement must be started by a rising edge at the START input when the segment to be measured starts, and stopped by a falling edge at the START input at the end of the segment. All values at the DATA output can be reset if a rising edge is generated at the RESET input at the same time as the rising edge at START. The measured values in the DATA structure that have already been determined then become invalid, and are re-calculated when the function block is called again.

Comment:
The determined times can deviate from the actual values, since already for the calls of the GETCPUCOUNTER function block some time is needed. This time depends on the particular computer, and is included in the determined times. Task interruptions, e.g. by the NC, are not detected and lead to longer measuring times.

Inputs

```
VAR_INPUT
    bStart : BOOL;
    bReset : BOOL;
END_VAR
```

Name	Type	Description
bStart	BOOL	A positive edge at this input starts the measurement of the execution time. A negative edge at this input stops the measurement, and causes the current, minimum, maximum and mean execution times to be recalculated. The variable stData.dwMeasureCycle [▶ 43] is incremented at the same time.
bReset	BOOL	All variables at the DATA output are reset if a rising edge is generated at this input at the same time as a rising edge at the START input. The old values for the current, minimum, maximum and mean execution times are reset, and are re-calculated for following measurements.

Outputs

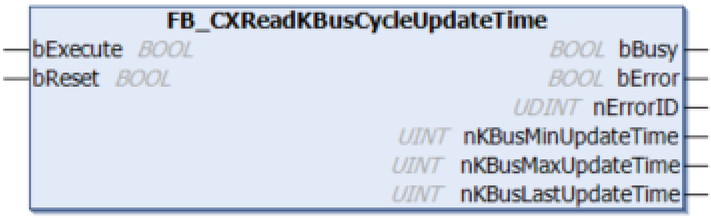
```
VAR_OUTPUT
    bBusy  : BOOL;
    stData : ST_CX_ProfilerStruct;
END_VAR
```

Name	Type	Description
bBusy	BOOL	This input is set at the start of the measuring procedure, and remains set until the time measurement has been completed. Once the bBusy output has been reset, the latest times are available at the stData output.
stData	ST_CX_ProfilerStruct	Structure of type ST_CX_ProfilerStruct [▶ 43] with the measured times [in µs].

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1	CX (WEC6/7: TC RT x86, WEC7: TC CE7 ARMV7)	Tc2_SystemCX

3.3 FB_CXReadKBusCycleUpdateTime



The function block can be used for all CXs that are operated directly with K-bus terminals. The function block determines the update time of the K-bus (K-bus runtime) with min. and max. values. The min. and max. values can be reset.

The K-bus runs task-synchronously with the PLC program. When the PLC program is completed, the K-bus is started, the outputs are written and the inputs are read. This means that a complete cycle always consists of the PLC runtime plus the K-bus update time.

- If the sum of both values is smaller than the set task cycle time, your system runs synchronously and thus optimally.
- If the sum is greater than your task cycle time, your system is no longer running in real time. It is recommended to avoid this state.
  You fix it by adjusting the task time, revising your PLC program or reducing the size of the K-bus.

Inputs

```
FUNCTION_BLOCK FB_CXReadKBusCycleUpdateTime
VAR_INPUT
    bExecute             : BOOL;        // rising edge triggers read process
    bReset               : BOOL;        // set TRUE to reset the min. and max. values
END_VAR
```

Name	Type	Description
bExecute	BOOL	A positive edge starts the function block.
bReset	BOOL	Reset of min. and max. values

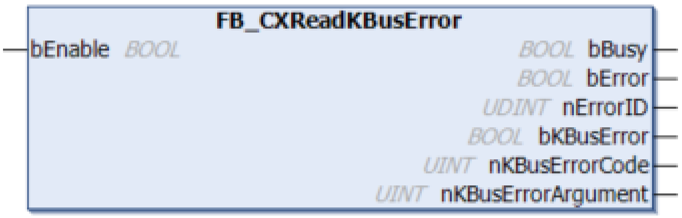
Outputs

```
VAR_OUTPUT
    bBusy                : BOOL;     // FB is in process
    bError               : BOOL;     // FB has an error
    nErrorID             : UDINT;    // ADS Error Code
    nKBusMinUpdateTime   : UINT;     // min. cycle update time in [µs]
    nKBusMaxUpdateTime   : UINT;     // max. cycle update time in [µs]
    nKBusLastUpdateTime  : UINT;     // last cycle update time in [µs]
END_VAR
```

Name	Type	Description
bBusy	BOOL	The function block is active and working.
bError	BOOL	The function block has an error.
nErrorID	UDINT	ADS error code
nKBusMinUpdateTime	UINT	Minimum update time in [µs] of the K-bus
nKBusMaxUpdateTime	UINT	Maximum update time in [µs] of the K-bus
nKBusLastUpdateTime	UINT	Last update time in [µs] of the K-bus

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4024.22	CX (x86, x64, ARM)	Tc2_SystemCX (System) >= 3.4.7.0

3.4 FB_CXReadKBusError



The function block enables further information to be read out about a K-bus error in order to obtain a more precise error image and to enable better diagnosis.

The prerequisite for this is that you use the K-bus terminals directly on your CX.

Example: If you are using the K-bus extension and the cable of the K-bus extension is disconnected, then you will see a K-bus interruption as `nErrorCode = 4` and the position where the K-bus has been interrupted as `nErrorArgument = Position`.

Further information on possible K-bus errors using the CX7000 as an example: https://infosys.beckhoff.com/content/1033/cx7000/9948355595.html?id=6787792405096234356

Inputs

Name	Type	Description
bEnable	BOOL	Link this variable to bit 0 of the K-bus state to activate the read process automatically in case of a K-bus error. The K-bus State is a Word variable. To mask out bit 0 of your linked variables, you can use `<VariableName>.0`.

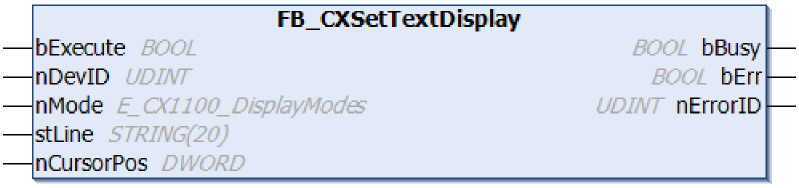
Outputs

Name	Type	Description
bBusy	BOOL	The function block is active and working.
bError	BOOL	The function block has an error.
nErrorID	UDINT	ADS error code
bKBusError	BOOL	The K-bus has an error. Further information is available at the outputs `nKBusErrorCode` and `nKBusErrorArgument`.
nKBusErrorCode	UINT	K-bus error code
nKBusErrorArgument	UINT	K-bus error argument

Once the error has been corrected, the K-bus can be restarted via the function block IOF_DeviceReset (from the PLC library Tc2_IoFunctions).

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4024.22	CX (x86, x64, ARM)	Tc2_SystemCX (System) >= 3.4.7.0

3.5 FB_CXSetTextDisplay



The functionblock FB_CXSetTextDisplay can be used to send text messages to the two line display of the CX1100.

Inputs

```
VAR_INPUT
    bExecute   : BOOL;
    nDevID     : UDINT;
    nMode      : E_CX1100_DisplayModes;
    stLine     : STRING(20);
    nCursorPos : DWORD;
END_VAR
```

Name	Type	Description
bExecute	BOOL	The command is executed with a rising edge.
nDevID	UDINT	Device ID of the CX1100 device.
nMode	E_CX1100_DisplayModes	Mode switching
stLine	STRING	String with 20 characters. This string is shown in the display with the corresponding command.
nCursorPos	DWORD	Cursor position. The string is written from this position in the display.

Outputs

```
VAR_OUTPUT
    bBusy     : BOOL;
    bErr      : BOOL;
    nErrorID : UDINT;
END_VAR
```

Name	Type	Description
bBusy	BOOL	The command is in the process of being transmitted by ADS. No new command will be accepted as long as `bBusy` remains TRUE.
bErr	BOOL	Becomes TRUE, as soon as an error occurs.
nErrorID	UDINT	Supplies the error number when the `bError` output is set.

E_CX1000_DisplayModes :

```
E_CX1000_DisplayModes : (
 e_CX1100_DisplayNoAction := 0,
 e_CX1100_DisplayOn := 1,
 e_CX1100_DisplayOff,
 e_CX1100_CursorOn,
 e_CX1100_CursorOff,
 e_CX1100_CursorBlinkOn,
 e_CX1100_CursorBlinkOff,
 e_CX1100_BackLightOn,
 e_CX1100_BackLightOff,
 e_CX1100_ClearDisplay,
 e_CX1100_WriteLine1,
 e_CX1100_WriteLine2
);
```

e_CX1100_DisplayNoAction: no action.

e_CX1100_DisplayOn: switch on the display.

e_CX1100_DisplayOff: switch off the display.

e_CX1100_CursorOn: switch on the cursor.

e_CX1100_CursorOff: switch off the cursor.

e_CX1100_CursorBlinkOn: switch on the cursor blinking.

e_CX1100_CursorBlinkOff: switch off the cursor blinking.

e_CX1100_BackLightOn: switch on the backlight.

e_CX1100_BackLightOff: switch off the backlight.

e_CX1100_ClearDisplay: clear display.

e_CX1100_WriteLine1: write the first line.

e_CX1100_WriteLine2: write the second line.

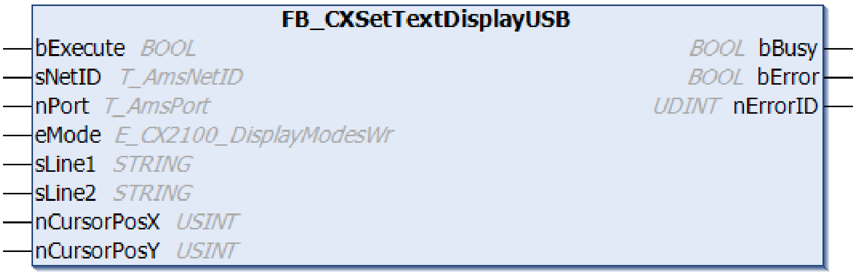
Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.0	CX (x86)	Tc2_SystemCX

3.6 FB_CXSetTextDisplayUSB

The function block is used for the CX2100 and for the EL6090 terminal. When the function block is called, only the matching NetID and port number for the respective device has to be allocated.

	nPort:	sNetID
CX2100	Is displayed in TwinCAT on the ESB device tab.	Is the NetID of the PC or is left blank (‘ ‘).
EL6090	Is the EtherCAT address of the terminal	Is the EtherCAT address of the EtherCAT master.



The function block FB_CXSetTextDisplayUSB is used to write and delete messages on the two-line display. The cursor is controlled on the display by switching it on and off or by making it flash. The function block is also used to switch the backlight on or off.

Inputs

```
VAR_INPUT
     bExecute      : BOOL;
     sNetID        : T_AmsNetID;
     nPort         : T_AmsPort;
     eMode         : E_CX2100_DisplayModesWr;
     sLine1        : STRING(80);
     sLine2        : STRING(80);
     nCursorPosX   : USINT;
     nCursorPosy   : USINT;
END_VAR
```

Name	Type	Description
bExecute	BOOL	The command is executed with a rising edge.
sNetID	T_AmsNetID	AMSNetID of the device
nPort	T_AmsPort	AmsPort of the device
eMode	E_CX2100_DisplayModesRd	Mode switching
sLine1	STRING	String with 80 characters. This string is displayed with the corresponding command in the first line. For strings with more than 16 characters, the text is displayed as scrolling text.
sLine2	STRING	String with 80 characters. This string is displayed with the corresponding command in the second line. For strings with more than 16 characters, the text is displayed as scrolling text.
nCursorPosX	USINT	Cursor position on the X axis. The string is written from this position in the display.
nCursorPosY	USINT	Cursor position on the Y axis. The string is written from this position in the display.

Outputs

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

Name	Type	Description
bBusy	BOOL	The command is in the process of being transmitted by ADS. No new command will be accepted as long as `bBusy` remains TRUE.
bError	BOOL	Becomes TRUE, as soon as an error occurs.
nErrorID	UDINT	Supplies the error number when the `bError` output is set.

E_CX2100_DisplayModesWr:

```
E_CX2100_DisplayModesWr : (
     eCX2100_DisplayNoActionWr := 0,
     eCX2100_CursorOn,
     eCX2100_CursorOff,
     eCX2100_CursorBlinkOn,
     eCX2100_CursorBlinkOff,
     eCX2100_BackLightOn,
     eCX2100_BackLightOff,
     eCX2100_ClearDisplay,
     eCX2100_WriteLine1,
     eCX2100_WriteLine2,
     eCX2100_WriteLines,
     eCX2100_CursorPosX,
     eCX2100_CursorPosY,
     eCX2100_CursorPosXY
;
```

eCX2100_DisplayNoActionWr: no action.

eCX2100_CursorOn: switch on the cursor.

eCX2100_CursorOff: switch off the cursor.

eCX2100_CursorBlinkOn: switch on the cursor blinking.

eCX2100_CursorBlinkOff: switch off the cursor blinking.

eCX2100_BackLightOn: switch on the backlight.

eCX2100_BackLightOff: switch off the backlight.

eCX2100_ClearDisplay: clear display.

eCX2100_WriteLine1: write the first line.

eCX2100_WriteLine2: write the second line.

eCX2100_WriteLines: write lines.

eCX2100_CursorPosX: cursor position on the X axis.

eCX2100_CursorPosY: cursor position on the Y axis.

eCX2100_CursorPosXY: cursor position on the XY axis.

Requirements when using the EL6090

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1	PC or CX (WES7/Win7/Win10: TC RT x86/x64, WEC6/7: TC RT x86, WEC7: TC CE7 ARMV7, TC/BSD: TC RT x64, TC OS ARMT2)	Tc2_SystemCX

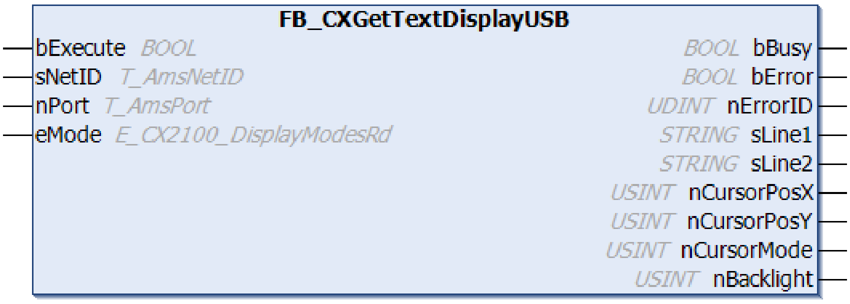
Requirements when using the CX2100

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1	CX (WES7/Win7/Win10: TC RT x86/x64, WEC6/7: TC RT x86, TC/BSD: TC RT x64)	Tc2_SystemCX

3.7 FB_CXGetTextDisplayUSB

The function block is used for the CX2100 and for the EL6090 terminal. When the function block is called, only the matching NetID and port number for the respective device has to be allocated.

	nPort:	sNetID
CX2100	Is displayed in TwinCAT on the ESB device tab.	Is the NetID of the PC or is left blank (‘ ‘).
EL6090	Is the EtherCAT address of the terminal	Is the EtherCAT address of the EtherCAT master.



The function block FB_CXGetTextDisplayUSB is used to read the lines on the display. In addition, the cursor status is read, i.e. cursor switched on or off, or is flashing. The function block also indicates whether the backlight is switched on or off.

Inputs

```
VAR_INPUT
     bExecute       : BOOL;
     sNetID         : T_AmsNetID;
     nPort          : T_AmsPort;
     eMode          : E_CX2100_DisplayModesRd;
END_VAR
```

Name	Type	Description
bExecute	BOOL	The command is executed with a rising edge.
sNetID	T_AmsNetID	AMSNetID of the device
nPort	T_AmsPort	AmsPort of the device
eMode	E_CX2100_DisplayModesRd	Mode switching

Outputs

```
VAR_OUTPUT
     bBusy          : BOOL;
     bError         : BOOL;
     nErrorID       : UDINT;
     sLine1         : STRING(80);
     sLine2         : STRING(80);
     nCursorPosX    : USINT;
     nCursorPosY    : USINT;
     nCursorMode    : USINT;
     nBacklight     : USINT;
END_VAR
```

Name	Type	Description
bBusy	BOOL	The command is in the process of being transmitted by ADS. No new command will be accepted as long as `bBusy` remains TRUE.
bError	BOOL	Becomes TRUE, as soon as an error occurs.
nErrorID	UDINT	Supplies the error number when the `bError` output is set.
sLine1	STRING	This string is read with the corresponding command.
sLine2	STRING	This string is read with the corresponding command.
nCursorPosX	USINT	Cursor position on the X axis
nCursorPosY	USINT	Cursor position on the Y axis
nCursorMode	USINT	Cursor mode
nBacklight	USINT	Backlight

E_CX2100_DisplayModesRd:

```
E_CX2100_DisplayModesRd : (
     eCX2100_DisplayNoActionRd := 0,
     eCX2100_ReadCursorInfo,
     eCX2100_ReadBackLight,
     eCX2100_ReadLine1,
     eCX2100_ReadLine2,
     eCX2100_ReadLines
);
```

eCX2100_DisplayNoActionRd: no action.

eCX2100_ReadCursorInfo: read values via the cursor.

eCX2100_ReadBackLight: read backlight values.

eCX2100_ReadLine1: read values from the first line.

eCX2100_ReadLine2: read values from the second line.

eCX2100_ReadLines: read values from lines.

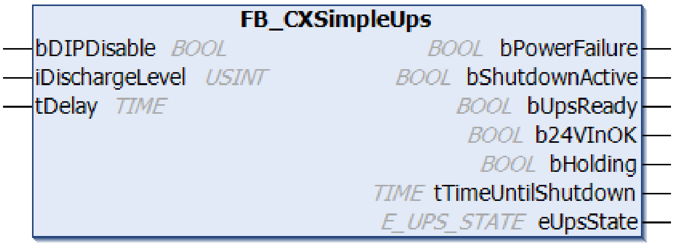
Requirements when using the CX2100

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1	CX (WES7/Win7/Win10: TC RT x86/x64, WEC6/7: TC RT x86, TC/BSD: TC RT x64)	Tc2_SystemCX

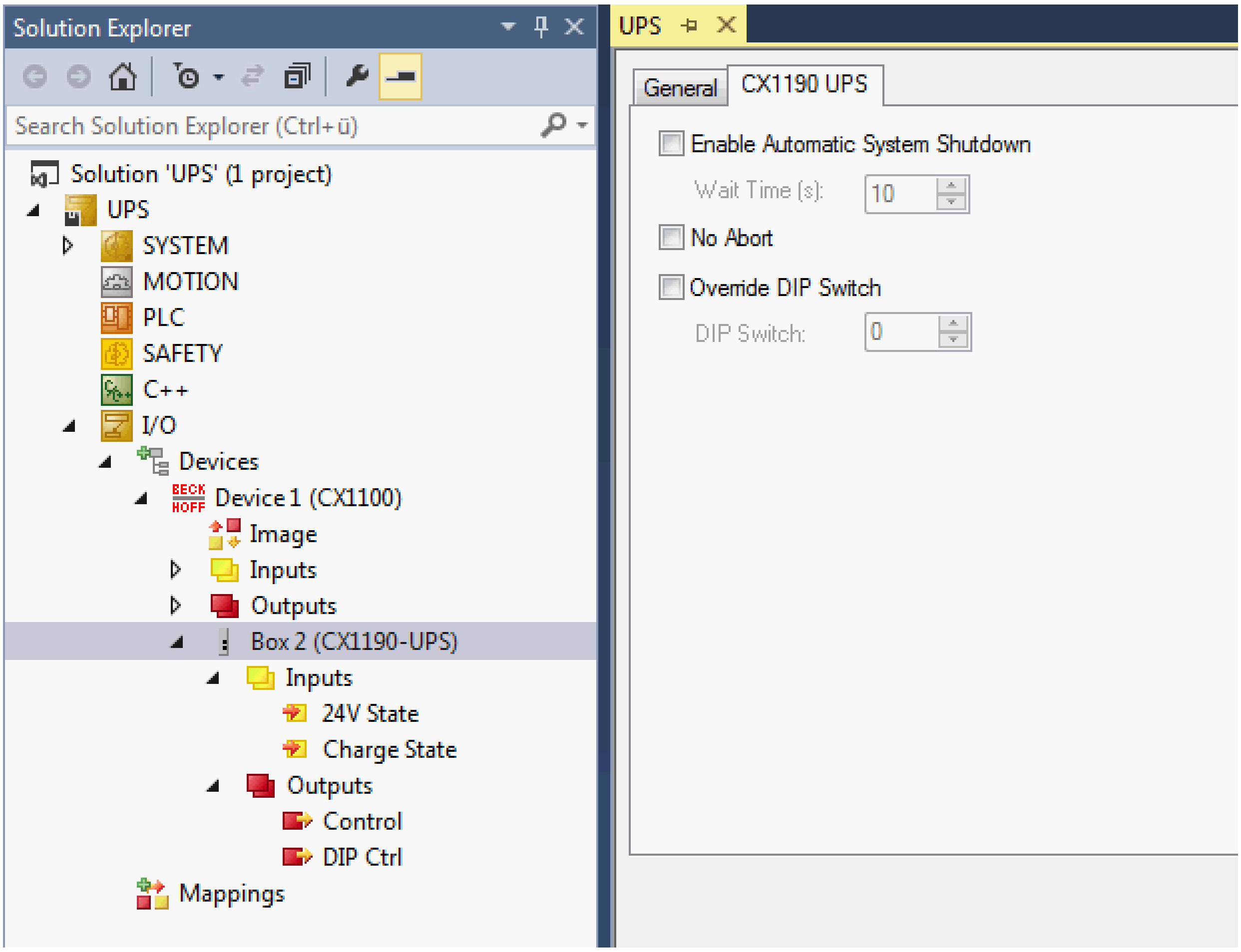
Requirements when using the EL6090

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1	PC or CX (WES7/Win7/Win10: TC RT x86/x64, WEC6/7: TC RT x86, WEC7: TC CE7 ARMV7, TC/BSD: TC RT x64, TC OS ARMT2)	Tc2_SystemCX

3.8 FB_CXSimpleUps



The function block FB_CXSimpleUps can be used on the CX1000 or CX1020 in order to activate the UPS CX1190-UPS from the PLC. In this case the UPS settings must be deactivated in the TwinCAT System Manager.



□ Enable Automatic System Shutdown

Wait Time (s): 10

□ No Abort

□ Override DIP Switch

DIP Switch: 0

Inputs

```
VAR_INPUT
    bDIPDisable     : BOOL;
    iDischargeLevel : USINT;
    tDelay          : TIME;
END_VAR
```

Name	Type	Description
bDIPDisable	BOOL	If bDIPDIsable = TRUE, then the position of the charge level switch on the UPS is ignored and iDischargeLevel is used instead.
iDischargeLevel	USINT	Discharge switch-off threshold: 0 = 100% (maximum discharge), 9 = 90%, 8 = 80%, ..., 2 = 20%, 1 = 10% (minimum discharge).
tDelay	TIME	Holding time before the shutdown is carried out. It is used to bridge short power failures (up to 10 s). After exceeding the holding time, the holding period is aborted. Internally the FB waits for 2.5 s. If the voltage has returned by then, the FB returns to normal operation, otherwise the system is shut down. If the voltage returns during or after shutdown, the CX automatically reboots after discharging and recharging the UPS.

Outputs

```
VAR_OUTPUT
    bPowerFailure      : BOOL;
    bShutdownActive    : BOOL;
    bUpsReady          : BOOL;
    b24VInOK           : BOOL;
    bHolding           : BOOL;
    tTimeUntilShutdown : TIME;
    eUpsState          : E_UPS_STATE;
END_VAR
```

Name	Type	Description
bPowerFailure	BOOL	Becomes TRUE when a power failure of the supply voltage is detected, becomes FALSE when the input voltage returns.
bShutdownActiv e	BOOL	Becomes TRUE as soon as the stop or shutdown is executed.
bUpsReady	BOOL	Becomes TRUE as soon as the 24 V input voltage is available.
b24VInOK	BOOL	Becomes TRUE as soon as the UPS provides the output voltage.
bHolding	BOOL	Becomes TRUE as soon as a failure in the supply voltage has been detected, and the holding time has not yet elapsed.
tTimeUntilShutd own	TIME	Indicates the holding time remaining until shutdown.
eUpsState	E_UPS_STATE	Displays the status of the UPS [UNDEF | CHARGING | CHARGED | DISCHARGE | DISCHARGE_RESTART | OUTPUT_OFF | OVERLOAD].

Configuration variables

```
VAR_CONFIG
    Ii24VState    AT %I* : BYTE;
    IiChargeState AT %I* : USINT;
    QiControl     AT %Q* : BYTE;
    QiDipControl  AT %Q* : USINT;
END_VAR
```

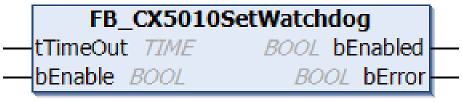
Name	Type	Description
Ii24VState	BYTE	Must be linked with input '24V State', see picture above.
IiChargeState	USINT	Must be linked with input 'Charge State', see picture above.
QiControl	BYTE	Must be linked with output 'Contol', see picture above.
QiDipControl	USINT	Must be linked with output 'DIP Ctrl', see picture above.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.0	CX (x86)	Tc2_SystemCX

For other CX and PC please refer to the function block FB_S_SUPS_BAPI in the Tc2_SUPS library.

3.9 FB_CX5010SetWatchdog



The function block FB_CX5010SetWatchdog activates a hardware watchdog on the CX5010. The watchdog is activated via bEnable = TRUE and the tTimeOut time. The tTimeOut time can be a minimum of 2 seconds and a maximum of 255 seconds.

Once the watchdog has been activated, the function block instance must be called cyclically at shorter intervals than tTimeOut, since the CX5010 restarts automatically when tTimeOut has elapsed. The watchdog can therefore be used to automatically reboot systems, which have entered an infinite loop or where the PLC has become stuck.

The watchdog can be deactivated via bEnable = FALSE or tTimeOut = T#0s.

NOTICE

The watchdog must be deactivated before breakpoints are used, before a PLC reset or an overall reset, before a TwinCAT stop, before switching to Config mode or before the configuration is activated, because otherwise the CX5010 would reboot immediately once tTimeOut has elapsed.

Inputs

```
VAR_INPUT
    tTimeOut : TIME;
    bEnable  : BOOL;
END_VAR
```

Name	Type	Description
tTimeOut	TIME	Watchdog time, after which a restart is performed.
bEnable	BOOL	Activating/deactivating the watchdog.

Outputs

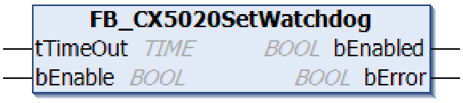
```
VAR_OUTPUT
    bEnabled : BOOL;
    bError   : BOOL;
END_VAR
```

Name	Type	Description
bEnabled	BOOL	TRUE = Watchdog is active, FALSE = Watchdog is not active.
bError	BOOL	Error when activating or deactivating the watchdog.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.0	CX (x86)	Tc2_SystemCX

3.10 FB_CX5020SetWatchdog



The function block FB_CX5020SetWatchdog activates a hardware watchdog on the CX5020. The watchdog is activated via bEnable = TRUE and the tTimeOut time. The tTimeOut time can be a minimum of 2 seconds and a maximum of 255 seconds.

Once the watchdog has been activated, the function block instance must be called cyclically at shorter intervals than tTimeOut, since the CX5020 restarts automatically when tTimeOut has elapsed. The watchdog can therefore be used to automatically reboot systems, which have entered an infinite loop or where the PLC has become stuck.

The watchdog can be deactivated via bEnable = FALSE or tTimeOut = T#0s.

NOTICE

The watchdog must be deactivated before breakpoints are used, before a PLC reset or an overall reset, before a TwinCAT stop, before switching to Config mode or before the configuration is activated, because otherwise the CX5020 would reboot immediately once tTimeOut has elapsed.

Inputs

```
VAR_INPUT
    tTimeOut : TIME;
    bEnable  : BOOL;
END_VAR
```

Name	Type	Description
tTimeOut	TIME	Watchdog time, after which a restart is performed.
bEnable	BOOL	Activating/deactivating the watchdog.

Outputs

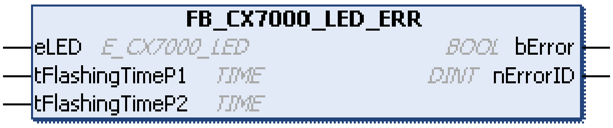
```
VAR_OUTPUT
    bEnabled : BOOL;
    bError   : BOOL;
END_VAR
```

Name	Type	Description
bEnabled	BOOL	TRUE = Watchdog is active, FALSE = Watchdog is not active.
bError	BOOL	Error when activating or deactivating the watchdog.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.0	CX (x86)	Tc2_SystemCX

3.11 FB_CX7000_LED_ERR



The function block allows the use of the ERR LED on the CX7000. The function block is immediately active when it is called and controls the ERR LED via the mode.

The ERR LED of the CX7000 can be used to make the states of the PLC program, communication or other indications externally visible.

The ERR LED has two colors, red and green. If both colors are switched on, the LED lights up yellow. You can either turn the LED on or make it flash.

User-specific function of the LEDs

Due to the user-specific usability of the LEDs, Beckhoff Support cannot know the meaning of a flashing code and cannot support the customer.

- Document the function of the LEDs for your customers.

Inputs

```
VAR_INPUT
    bEnable             : BOOL;             // set TRUE to enable LED handling; Reset in order to re
set error
    eLED                : E_CX7000_LED;     // LED flashing mode
    tFlashingTimeP1     : TIME:=T#250MS;    // Flashing Time >=200ms first pulse
    tFlashingTimeP2     : TIME:=T#250MS;    // Flashing Time >=200ms second pulse
END_VAR
```

Name	TYPE	Description
bEnable	BOOL	The function block controls the LED as soon as and as long as the input is TRUE.
eLED	E_CX7000_LED	LED mode
tFlashingTimeP1	TIME	Time for the first pulse (>= 200 ms)
tFlashingTimeP2	TIME	Time for the second pulse (>= 200 ms)

Outputs

```
VAR_OUTPUT
    bError          : BOOL;        // error flag
    nErrorID        : UDINT;       (* ADS Error ID. If nErrorID=DEVICE_SRVNOTSUPP probably the image
 version need to be updated to support this feature. *)
END_VAR
```

Name	Type	Description
bError	BOOL	The function block has an error.
nErrorID	UDINT	ADS Error Code Example: DEVICE_SRVNOTSUPP: the image version of the CX7000 does not support this feature. An update (>=35695) is necessary.

Sample:

```
VAR
    BK9000_BoxState AT %I* : WORD;
    fbErrorLED : FB_CX7000_LED_ERR;
END_VAR
```

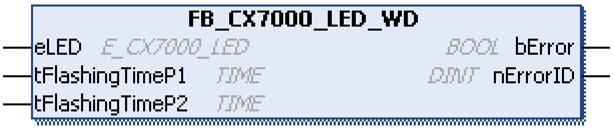
```
IF BK9000_BoxState=0 THEN
fbErrorLED.eLED :=E_CX7000_LED.LED_flashing_GREEN_OFF;
ELSE
fbErrorLED.eLED :=E_CX7000_LED.LED_flashing_RED_OFF;
END_IF
fbErrorLED(
    bEnable := TRUE,
    tFlashingTimeP1 := ,
    tFlashingTimeP2 := ,
    bError => ,
    nErrorID => );
```

NOTICE

Function block can only be used for CX7000

The function block can and must only be used for the CX7000.

3.12 FB_CX7000_LED_WD



The function block allows the use of the WD LED on the CX7000. The function block is immediately active with the call and controls the WD LED via the mode.

You can use the WD LED of the CX7000 to make the states of the PLC program, communication or other indications externally visible. The WD LED has two colors, red and green. If both colors are switched on, the LED lights up yellow. You can turn on the LED and/or make it flash.

User-specific function of the LEDs

Due to the user-specific usability of the LEDs, Beckhoff Support cannot know the meaning of a flashing code and cannot support the customer.

- Document the function of the LEDs for your customers.

Inputs

```
VAR_INPUT
    bEnable             : BOOL;            // set TRUE to enable LED handling; Reset in order to re
set error
    eLED                : E_CX7000_LED;    // LED flashing mode
    tFlashingTimeP1     : TIME:=T#250MS;    // Flashing Time >=200ms first pulse
    tFlashingTimeP2     : TIME:=T#250MS;    // Flashing Time >=200ms second pulse
END_VAR
```

Name	Type	Description
bEnable	BOOL	The function block controls the LED as soon as and as long as the input is TRUE.
eLED	E_CX7000_LED	LED mode
tFlashingTimeP1	TIME	Time for the first pulse (>= 200 ms)
tFlashingTimeP2	TIME	Time for the second pulse (>= 200 ms)

Outputs

```
VAR_OUTPUT
    bError          : BOOL;        // error flag
    nErrorID        : UDINT;       (* ADS Error ID. If nErrorID=DEVICE_SRVNOTSUPP probably the image v
ersion need to be updated to support this feature. *)
END_VAR
```

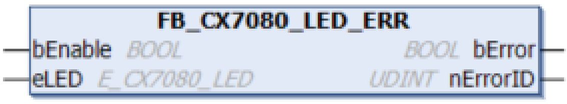
Name	Type	Description
bError	BOOL	The function block has an error.
nErrorID	UDINT	ADS Error Code Example: DEVICE_SRVNOTSUPP: the image version of the CX7000 does not support this feature. An update (>=35695) is necessary.

NOTICE

Function block can only be used for CX7000

The function block can and must only be used for the CX7000.

3.13 FB_CX7080_LED_ERR



You can use the WD/ERR LEDs of the CX7080 to make the states of the PLC program, communication or other indications externally visible.

The function block enables the ERR LED of the CX7080 to be set. The LED has two colors, red and green. If both colors are switched on, the LED lights up yellow. You therefore have three colors at your disposal. You can turn on the LED and/or make it flash.

To use the LED, the corresponding COM port must be included in the configuration. The RS485 interface is linked to the ERR LED.

User-specific function of the LEDs

Due to the user-specific usability of the LEDs, Beckhoff Support cannot know the meaning of a flashing code and cannot support the customer.

- Document the function of the LEDs for your customers.

Inputs

```
VAR_INPUT
    bEnable     : BOOL;                 // set TRUE to enable LED handling; Reset in order to reset e
rror
    eLED        : E_CX7080_LED;         // LED flashing mode
END_VAR
```

Name	Type	Description
bEnable	BOOL	The function block controls the LED as soon as and as long as the input is TRUE.
eLED	E_CX7080_LED	LED mode

Outputs

```
VAR_OUTPUT
    bError      : BOOL;              // error flag
    nErrorID    : UDINT;             (* ADS Error ID.
If nErrorID=DEVICE_NOTFOUND probably the COM port is not set in the TC config.
If nErrorID=DEVICE_SRVNOTSUPP probably the image version need to be updated to support this feature.
*)
END_VAR
```

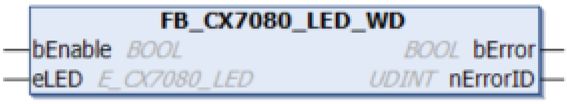
Name	Type	Description
bError	BOOL	The function block has an error.
nErrorID	UDINT	ADS Error Code Examples: DEVICE_NOTFOUND: probably the COM port is not included in the TwinCAT system configuration. DEVICE_SRVNOTSUPP: the image version of the CX7080 does not support this feature. An update (>=35695) is necessary.

NOTICE

Function block can only be used for CX7080

The function block can and must only be used for the CX7080.

3.14 FB_CX7080_LED_WD



You can use the WD/ERR LEDs of the CX7080 to make the states of the PLC program, communication or other indications externally visible.

The function block enables the WD LED of the CX7080 to be set. The LED has two colors, red and green. If both colors are switched on, the LED lights up yellow. You therefore have three colors at your disposal. You can turn on the LED and/or make it flash.

To use the LED, the corresponding COM port must be included in the configuration. The RS232 interface is linked to the WD LED.

User-specific function of the LEDs

Due to the user-specific usability of the LEDs, Beckhoff Support cannot know the meaning of a flashing code and cannot support the customer.

- Document the function of the LEDs for your customers.

Inputs

```
VAR_INPUT
    bEnable     : BOOL;                 // set TRUE to enable LED handling; Reset in order to reset e
rror
    eLED        : E_CX7080_LED;         // LED flashing mode
END_VAR
```

Name	Type	Description
bEnable	BOOL	The function block controls the LED as soon as and as long as the input is TRUE.
eLED	E_CX7080_LED	LED mode

Outputs

```
VAR_OUTPUT
    bError      : BOOL;              // error flag
    nErrorID    : UDINT;             (* ADS Error ID.
If nErrorID=DEVICE_NOTFOUND probably the COM port is not set in the TC config.
If nErrorID=DEVICE_SRVNOTSUPP probably the image version need to be updated to support this feature.
*)
END_VAR
```

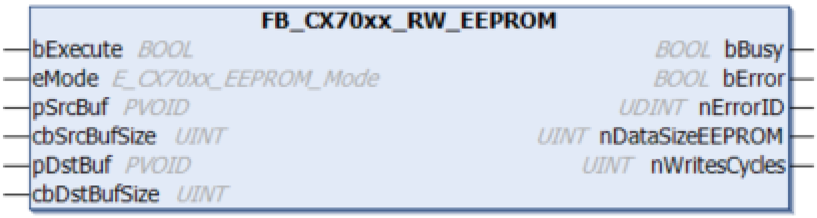
Name	Type	Description
bError	BOOL	The function block has an error.
nErrorID	UDINT	ADS Error Code Examples: DEVICE_NOTFOUND: probably the COM port is not included in the TwinCAT system configuration. DEVICE_SRVNOTSUPP: the image version of the CX7080 does not support this feature. An update (>=35695) is necessary.

NOTICE

Function block can only be used for CX7080

The function block can and must only be used for the CX7080.

3.15 FB_CX70xx_RW_EEPROM



The function block allows a maximum of 120 bytes to be written to the EEPROM (hardware) of the CX70xx. The EEPROM may be written to a maximum of 200 times. The memory is intended for one-time writing.

This function block can be used to personalize the CX70xx. That means, in the simplest case you write your company ID into the EEPROM. When starting the CX70xx program, read the contents of the memory. For example, if it is empty, you cannot continue to run the program because it is no longer your original CX70xx that you programmed.

If you want to exchange a CX70xx for a new device, the EEPROM must be written again by you.

Inputs

```
VAR_INPUT
    bExecute        : BOOL;         // rising edge triggers process with selected mode
    eMode           : E_CX70xx_EEPROM_Mode;    // select RW mode
    pSrcBuf         : PVOID;        // pointer to WRITE EEPROM data buffer
    cbSrcBufSize    : UINT;         // size of WRITE EEPROM data buffer (max.120 Bytes)
    pDstBuf         : PVOID;        // pointer to READ EEPROM data buffer
    cbDstBufSize    : UINT;         // max.size of READ EEPROM data buffer (max.120 Bytes)
END_VAR
```

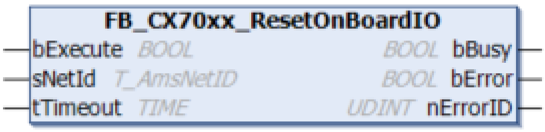
Name	Type	Description
bExecute	BOOL	A positive edge starts the function block.
eMode	E_CX70xx_EEPROM_Mode	ReadOnly: EEPROM read WriteOnly: EEPROM write WriteAndRead: EEPROM write and read
pSrcBuf	PVOID	Pointer to the data buffer to be written.
cbSrcBufLen	UINT	Length of data to be written (max. 120 bytes)
pDstBuf	PVOID	Pointer to the data buffer into which the contents of the EEPROM are to be copied.
cbDstBufLen	UINT	Length of data to be read. (maximum 120 bytes) When reading, the length information must be greater than or equal to the data contained in the EEPROM.

Outputs

```
VAR_OUTPUT
    bBusy            : BOOL;         // FB is working
    bError           : BOOL;         // FB has an Error
    nErrorID         : UDINT;        (* Error Code
    If nErrorID=DEVICE_INVALIDACCESS the EEPROM write cycles reached max. value.
    If nErrorID=DEVICE_INVALIDPARM the given pointer parameter is invalid/null.
    If nErrorID=DEVICE_INVALIDSIZE the given buffer size is too small or too big.
    If nErrorID=DEVICE_SRVNOTSUPP probably the image version need to be updated to support this feat
ure. *)
    nDataSizeEEPROM  : UINT;         // current size of (read) EEPROM data in bytes (max.120 Bytes)
    nWritesCycles    : UINT;         // already performed EEPROM write cycles (maximum possible = 20
0)
END_VAR
```

Name	Type	Description
bBusy	BOOL	The function block is active and working.
bError	BOOL	The function block has an error.
nErrorID	UDINT	ADS Error Code Examples: DEVICE_INVALIDACCESS: the EEPROM write cycles have reached the maximum value. The EEPROM cannot be rewritten. DEVICE_INVALIDPARM: the allocated pointers are invalid/NULL. DEVICE_INVALIDSIZE: the allocated buffer size is too small or too large. DEVICE_SRVNOTSUPP: the image version of the CX70xx does not support this feature. An update (>=35695) is necessary.
nDataSizeEEPROM	UINT	Current size in bytes of the read EEPROM data
nWritesCycles	UINT	Number of write operations still available

3.16 FB_CX70xx_ResetOnBoardIO



The function block allows to execute a reset from the OnBoard I/O of the CX70xx Embedded PC.

Typical use case is after an error in the communication to the OnBoard I/Os (CX7028). Such an error occurs when the power supply (Up) of the OnBoard I/Os is interrupted.

NOTICE

State of the I/Os

Outputs that are still set in the process image are switched on again immediately after a reset.

Further details on the OnBoard I/O can be found in the documentation of the CX70xx Embedded PC.

Inputs

```
VAR_INPUT
    bExecute      : BOOL;         // rising edge triggers process
    sNetId        : T_AmsNetID;   // AMS Net ID of the OnBoard IOs
    tTimeout      : TIME := DEFAULT_ADS_TIMEOUT; // maximum time allowed for execution of this ADS c
ommand
END_VAR
```

Name	Type	Description
bExecute	BOOL	A positive edge starts the function block.
sNetId	T_AmsNetID	AMS Net ID of the OnBoard I/Os
tTimeout	TIME	States the length of the timeout that may not be exceeded by execution of the ADS command.

Outputs

```
VAR_OUTPUT
    bBusy       : BOOL;       // FB is working
    bError      : BOOL;       // FB has an Error
    nErrorID    : UDINT;      (* Error Code. If nErrorID=DEVICE_SRVNOTSUPP probably the image versio
n need to be updated to support this feature. *)
END_VAR
```

Name	Type	Description
bBusy	BOOL	The function block is active and working.
bError	BOOL	The function block has an error.
nErrorID	UDINT	ADS Error Code Examples: DEVICE_SRVNOTSUPP: the image version of the CX70xx does not support this feature. An update (>=47912) is necessary.

Sample:

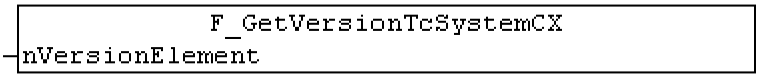
```
FUNCTION_BLOCK FB_Test_ResetOnboardIO
VAR
    AMSNetID     : T_AmsNetIdArr;     // link to the AMS Net ID of the OnBoard IOs
    State        : WORD;              // link to the State of the OnBoard IOs
    bReset       : BOOL;              // if Ready to Reset you can reset the OnBoard IOs
    fbReset      : FB_CX70xx_ResetOnBoardIO;
END_VAR
```

```
IF State<>8 AND NOT State.8 AND State.4 THEN // if OnBoard IO device signals an error and is not OP
but present
    bReset := TRUE;
ELSE
    bReset := FALSE;
END_IF

IF NOT fbReset.bBusy AND bReset THEN
    fbReset(bExecute:=TRUE, sNetId:=F_CreateAmsNetId(AMSNetID));
ELSE
    fbReset(bExecute:=FALSE);
END_IF
```

4 Functions

4.1 [obsolete]

4.1.1 F_GetVersionTcCXSystem



The function returns library version info.

FUNCTION F_GetVersionTcSystemCX : UINT

```
VAR_INPUT
nVersionElement : INT;
END_VAR
```

nVersionElement : Version element:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

4.1.2 F_GetVersionTcSystemCX1000

F_GetVersionTcSystemCX1000
nVersionElement

The function returns library version info.

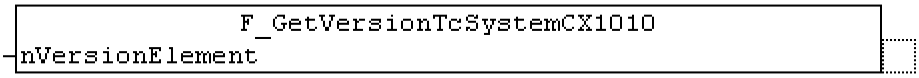
FUNCTION F_GetVersionTcSystemCX1000 : UINT

```
VAR_INPUT
nVersionElement : INT;
END_VAR
```

nVersionElement : Version element:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

4.1.3 F_GetVersionTcSystemCX1010



The function returns library version info.

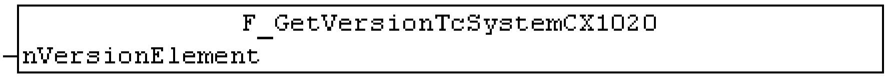
FUNCTION F_GetVersionTcSystemCX1010 : UINT

```
VAR_INPUT
nVersionElement : INT;
END_VAR
```

nVersionElement : Version element:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

4.1.4 F_GetVersionTcSystemCX1020



The function returns library version info.

FUNCTION F_GetVersionTcSystemCX1020 : UINT

```
VAR_INPUT
nVersionElement : INT;
END_VAR
```

nVersionElement : Version element:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

4.1.5 F_GetVersionTcSystemCX1030

The function returns library version info.

FUNCTION F_GetVersionTcSystemCX1030 : UINT

```
VAR_INPUT
nVersionElement : INT;
END_VAR
```

nVersionElement : Version element:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

4.1.6 F_GetVersionTcSystemCX5010

The function returns library version info.

FUNCTION F_GetVersionTcSystemCX5010 : UINT

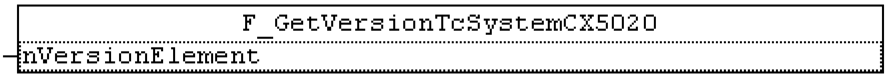
```
VAR_INPUT
nVersionElement : INT;
END_VAR
```

nVersionElement : Version element:

- 1 : major number;

- 2 : minor number;
- 3 : revision number;

4.1.7 F_GetVersionTcSystemCX5020



The function returns library version info.

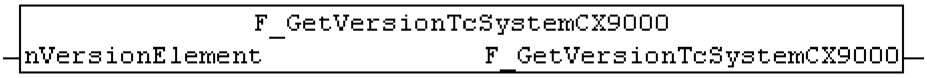
FUNCTION F_GetVersionTcSystemCX5020 : UINT

```
VAR_INPUT
nVersionElement : INT;
END_VAR
```

nVersionElement : Version element:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

4.1.8 F_GetVersionTcSystemCX9000



The function returns library version info.

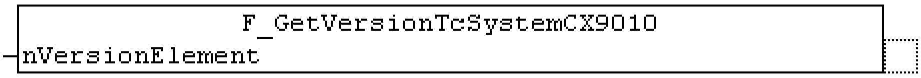
FUNCTION F_GetVersionTcSystemCX9000 : UINT

```
VAR_INPUT
nVersionElement : INT;
END_VAR
```

nVersionElement : Version element:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

4.1.9 F_GetVersionTcSystemCX9010



The function returns library version info.

FUNCTION F_GetVersionTcSystemCX9010 : UINT

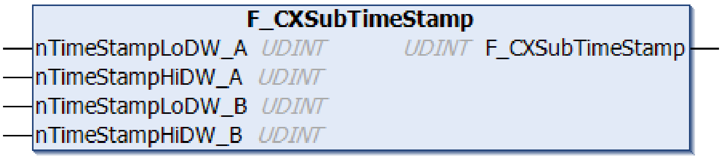
```
VAR_INPUT
nVersionElement : INT;
END_VAR
```

nVersionElement : Version element:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

4.1.10 F_CXSubTimeStamp

For subtraction TwinCAT3.1 also offers 64-bit data types (LINT/ULINT, LWORD) that can be executed directly (A-B) or there are TC2-compatible functions for 64-bit operations that should be used as an alternative to F_CXSubTimeStamp.



The function F_CXSubTimeStamp executes a 64-bit subtraction time stamp A - time stamp B and converts the result to µs. The required 64-bit time stamps with 100 ns resolution can be read from the system with the function block GETCPUCOUNTER.

If the difference between time stamp A and time stamp B is negative or greater than 4294967295 us, the maximum value 4294967295 us is returned. This corresponds to 71 minutes, 34 seconds, 967 milliseconds and 295 microseconds. In such cases the function UInt64Sub64() from TcUtilities.lib can be used to execute a complete 64-bit subtraction with 64-bit result in [100 ns].

FUNCTION F_CXSubTimeStamp: UDINT

```
VAR_INPUT
    nTimeStampLoDW_A : UDINT; (* 2*32 bit time stamp A: low DWORD *)
    nTimeStampHiDW_A : UDINT; (* 2*32 bit time stamp A: high DWORD *)
    nTimeStampLoDW_B : UDINT; (* 2*32 bit time stamp B: low DWORD *)
    nTimeStampHiDW_B : UDINT; (* 2*32 bit time stamp B: high DWORD *)
END_END_VAR
```

nTimeStampLoDW_A: lower 32bit of time stamp A.

nTimeStampHiDW_A: upper 32bit of time stamp A.

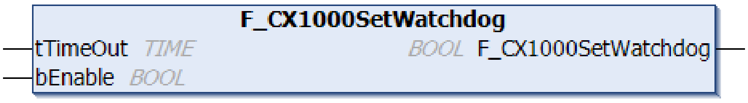
nTimeStampLoDW_B: lower 32bit of time stamp B.

nTimeStampHiDW_B: upper 32bit of time stamp B.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1	CX (WES7/Win7/Win10: TC RT x86/x64, WEC6/7 :TC TR x86, WEC7: TC CE7 ARMV7)	Tc2_SystemCX

4.1.11 F_CX1000SetWatchdog



The function F_CX1000SetWatchdog activates a hardware watchdog on the CX1000. The watchdog is activated via bEnable = TRUE and the tTimeout time. The tTimeout time can be a minimum of several PLC task cycles (multiple of the call time of the function F_CX1000SetWatchdog) and a maximum of 65 s and 535 ms.

Once the watchdog has been activated, the function must be called cyclically at shorter intervals than tTimeOut, since the CX1000 restarts automatically when tTimeOut has elapsed. The watchdog can therefore be used to automatically reboot systems, which have entered an infinite loop or where the PLC has become stuck.

The watchdog can be deactivated via bEnable = FALSE or tTimeOut = 0.

> ***NOTICE***
>
> The watchdog must be deactivated before breakpoints are used, before a PLC reset or an overall reset, before a TwinCAT stop, before switching to Config mode or before the configuration is activated, because otherwise the CX1000 would reboot immediately once the timeout has elapsed.

FUNCTION F_CX1000SetWatchdog: BOOL

```
VAR_INPUT
    tTimeout : TIME;
    bEnable  : BOOL;
END_VAR
```

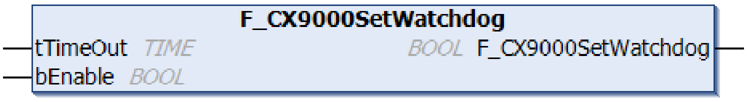
tTimeOut: Watchdog time, if expired a reboot is automatically executed.

bEnable: Activate or deactivate the watchdog.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.0	CX (x86)	Tc2_SystemCX

4.1.12 F_CX9000SetWatchdog



The function F_CX9000SetWatchdog activates a hardware watchdog on the CX9000. The watchdog is activated via bEnable = TRUE and the tTimeOut time. The tTimeOut time can be a maximum of 65 seconds and 535 milliseconds.

Once the watchdog has been activated, the function must be called cyclically at shorter intervals than tTimeOut, since the CX9000 restarts automatically when tTimeOut has elapsed. The watchdog can therefore be used to automatically reboot systems, which have entered an infinite loop or where the PLC has become stuck.

The watchdog can be deactivated via bEnable = FALSE or tTimeOut = T#0s.

> ***NOTICE***
>
> The watchdog must be deactivated before breakpoints are used, before a PLC reset or an overall reset, before a TwinCAT stop, before switching to Config mode or before the configuration is activated, because otherwise the CX9000 would reboot immediately once tTimeOut has elapsed.

FUNCTION F_CX9000SetWatchdog: BOOL

```
VAR_INPUT
    tTimeout : TIME;
    bEnable  : BOOL;
END_VAR
```

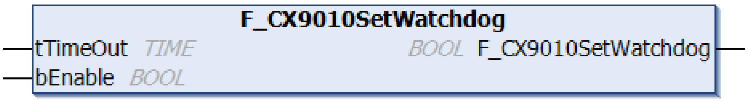
tTimeOut: Watchdog time, if expired a reboot is automatically executed.

bEnable: Activate or deactivate the watchdog.

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1.0	CX (ARM)	Tc2_SystemCX

4.1.13 F_CX9010SetWatchdog



The function F_CX9010SetWatchdog activates a hardware watchdog on the CX9000. The watchdog is activated via bEnable = TRUE and the tTimeOut time. The tTimeOut time can be a maximum of 65 seconds and 535 milliseconds.

Once the watchdog has been activated, the function must be called cyclically at shorter intervals than tTimeOut, since the CX9010 restarts automatically when tTimeOut has elapsed. The watchdog can therefore be used to automatically reboot systems, which have entered an infinite loop or where the PLC has become stuck.

The watchdog can be deactivated via bEnable = FALSE or tTimeOut = T#0s.

NOTICE

The watchdog must be deactivated before breakpoints are used, before a PLC reset or an overall reset, before a TwinCAT stop, before switching to Config mode or before the configuration is activated, because otherwise the CX9010 would reboot immediately once tTimeOut has elapsed.

FUNCTION F_CX9010SetWatchdog: BOOL

```
VAR_INPUT
    tTimeout : TIME;
    bEnable  : BOOL;
END_VAR
```

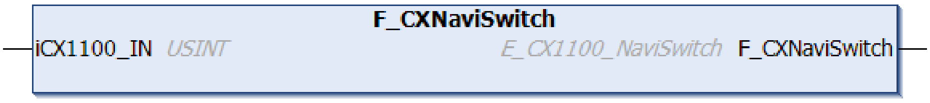
tTimeOut: Watchdog time, if expired a reboot is automatically executed.

bEnable: Activate or deactivate the watchdog.

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1.0	CX (ARM)	Tc2_SystemCX

4.2 F_CXNaviSwitch



The function F_CXNaviSwitch converts the value of the CX1100 navigation switch into an enum value of type E_CX1100_NaviSwitch.

FUNCTION F_CXNaviSwitch: E_CX1100_NaviSwitch

```
VAR_INPUT
    iCX1100_IN : USINT
END_VAR
```

E_CX1100_NaviSwitch: Value of the CX1100 input 'IN'

Enum E_CX1100_NaviSwitch

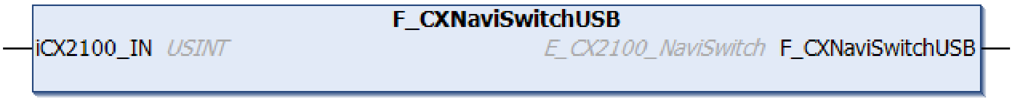
```
TYPE E_CX1100_NaviSwitch : (
    e_CX1100_NaviSwitch_IDLE := 0,
    e_CX1100_NaviSwitch_MIDDLE := 16,

    (* clockwise in 45 degree steps *)
    e_CX1100_NaviSwitch_TOP := 1,
    e_CX1100_NaviSwitch_TOPRIGHT := 9,
    e_CX1100_NaviSwitch_RIGHT := 8,
    e_CX1100_NaviSwitch_BOTTOMRIGHT := 10,
    e_CX1100_NaviSwitch_BOTTOM := 2,
    e_CX1100_NaviSwitch_BOTTOMLEFT := 6,
    e_CX1100_NaviSwitch_LEFT := 4,
    e_CX1100_NaviSwitch_TOPLEFT := 5,

    (* clockwise in 45 degree steps with middle switch pressed *)
    e_CX1100_NaviSwitch_MIDDLE_TOP := 17,
    e_CX1100_NaviSwitch_MIDDLE_TOPRIGHT := 25,
    e_CX1100_NaviSwitch_MIDDLE_RIGHT := 24,
    e_CX1100_NaviSwitch_MIDDLE_BOTTOMRIGHT := 26,
    e_CX1100_NaviSwitch_MIDDLE_BOTTOM := 18,
    e_CX1100_NaviSwitch_MIDDLE_BOTTOMLEFT := 22,
    e_CX1100_NaviSwitch_MIDDLE_LEFT := 20,
    e_CX1100_NaviSwitch_MIDDLE_TOPLEFT := 21
END_VAR
```

Values other than those defined in enum (e.g. 11) are displayed as "*** INVALID: value ***" in online mode (e.g. "*** INVALID: 11 ***"). The function F_CXNaviSwitch then returns the invalid value (e.g. 11).

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.0	CX (x86)	Tc2_SystemCX

4.3 F_CXNaviSwitchUSB



The function F_CXNaviSwitchUSB converts the value of the CX2100 navigation switch or an EL6090 into an enum value of type E_CX2100_NaviSwitch.

FUNCTION F_CXNaviSwitchUSB: E_CX2100_NaviSwitch

```
VAR_INPUT
    icx2100_IN : USINT;
END_VAR
```

icx2100_IN: Value of the CX2100 input 'IN‘

Enum E_CX2100_NaviSwitch

```
TYPE E_CX2100_NaviSwitch : (
    e_CX2100_NaviSwitch_IDLE  := 0,
    e_CX2100_NaviSwitch_MIDDLE  := 15,
    e_CX2100_NaviSwitch_ENTER  := 16,
    e_CX2100_NaviSwitch_ENTER_MIDDLE  := 31,

    (* clockwise in 45 degree steps, 1-2 switches on *)
    e_CX2100_NaviSwitch_TOP  := 1,
    e_CX2100_NaviSwitch_TOPRIGHT  := 9,
    e_CX2100_NaviSwitch_RIGHT  := 8,
    e_CX2100_NaviSwitch_BOTTOMRIGHT  := 10,
    e_CX2100_NaviSwitch_BOTTOM  := 2,
    e_CX2100_NaviSwitch_BOTTOMLEFT  := 6,
    e_CX2100_NaviSwitch_LEFT  := 4,
    e_CX2100_NaviSwitch_TOPLEFT  := 5,

    (* clockwise in 90 degree, 3 switches on*)
    e_CX2100_NaviSwitch_TOPLEFTRIGHT  := 13,
    e_CX2100_NaviSwitch_RIGHTTOPBOTTOM  := 11,
    e_CX2100_NaviSwitch_BOTTOMLEFTRIGHT  := 14,
    e_CX2100_NaviSwitch_LEFTTOPBOTTOM  := 7,

    (* clockwise in 45 degree steps with enter switch pressed, 1-2 switches on *)
    e_CX2100_NaviSwitch_ENTER_TOP  := 17,
    e_CX2100_NaviSwitch_ENTER_TOPRIGHT  := 25,
    e_CX2100_NaviSwitch_ENTER_RIGHT  := 24,
    e_CX2100_NaviSwitch_ENTER_BOTTOMRIGHT  := 26,
    e_CX2100_NaviSwitch_ENTER_BOTTOM  := 18,
    e_CX2100_NaviSwitch_ENTER_BOTTOMLEFT  := 22,
    e_CX2100_NaviSwitch_ENTER_LEFT  := 20,
    e_CX2100_NaviSwitch_ENTER_TOPLEFT  := 21
    (* clockwise in 90 degree steps with enter switch pressed, 3 switches on *)
    e_CX2100_NaviSwitch_ENTER_TOPLEFTRIGHT  := 29,
    e_CX2100_NaviSwitch_ENTER_RIGHTTOPBOTTOM  := 27,
    e_CX2100_NaviSwitch_ENTER_BOTTOMLEFTRIGHT  := 30,
    e_CX2100_NaviSwitch_ENTER_LEFTTOPBOTTOM  := 23,
)
```

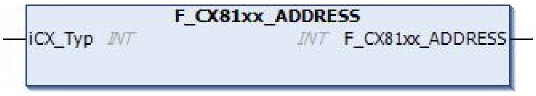
Requirements when using the CX2100

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1	CX (WES7/Win7/Win10: TC RT x86/x64, WEC6/7: TC RT x86, TC/BSD: TC RT x64)	Tc2_SystemCX

Requirements when using the EL6090

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1	PC or CX (WES7/Win7/Win10: TC RT x86/x64, WEC6/7: TC RT x86, WEC7: TC CE7 ARMV7, TC/BSD: TC RT x64, TC OS ARMT2)	Tc2_SystemCX

4.4 F_CX81xx_ADDRESS



This function reads the position of the address selection switch of the CXxxxx. One possible application is that you can activate different program parts in the PLC depending on the switch position.

This function reads the position of the DIP switch of the CXxxxx. One possible application is that you can activate different program parts in the PLC depending on the switch position.

VAR_INPUT

```
VAR_INPUT
    iCX_Typ      : INT;          (* Use product code without 'CX' e.g.: CX8180 -> 8180 *)
END_VAR
```

VAR_OUTPUT

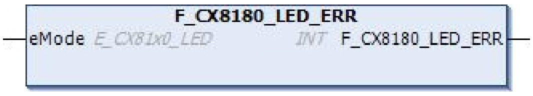
```
F_CX80xx_ADDRESS     : INT;
```

F_CX80xx_ADDRESS : -1, non-implemented CX, address of the switch

Requirements

Development environment	Target platform	Hardware	PLC libraries to include
TwinCAT v3.1 Build 4022.30	ARM	CXxxxx	Tc2_SystemCX

4.5 F_CX8180_LED_ERR



Since the CX8180 supports various protocols, the two LEDs WD and ERR on the CX8180 are not used by the firmware. This allows the user to create his own diagnosis messages. The LEDs can be used to indicate, for example, whether the CX8180 has received an IP address from the DHCP server or whether devices are exchanging data.

The F_CX8180_LED_ERR function controls the ERR LED on the CX8180. Various color and flashing modes can be used here. The possible LED colors are red and green.

VAR_INPUT

```
VAR_INPUT
    eMode      : E_CX81x0_LED;
END_VAR
```

eMode: The way in which the LED lights up, see also Data type E_CX81x0_LED [▸ 44].

Return Value

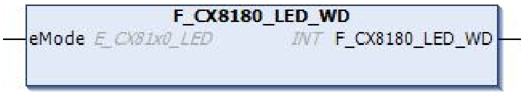
```
F_ F_CX8180_LED_ERR     : INT;
```

F_CX8180_LED_ERR: -1, not implemented flash code, 0 OK

Requirements

Development environment	Target platform	Hardware	PLC libraries to include
TwinCAT v3.1 Build 4022.30	ARM	CXxxxx	Tc2_SystemCX

4.6 F_CX8180_LED_WD



Since the CX8180 supports various protocols, the two LEDs WD and ERR on the CX8180 are not used by the firmware. This allows the user to create his own diagnosis messages. The LEDs can be used to indicate, for example, whether the CX8180 has received an IP address from the DHCP server or whether devices are exchanging data.

The F_CX8180_LED_WD function controls the WD LED on the CX8180. Various color and flashing modes can be used here. The possible LED colors are red and green.

VAR_INPUT

```
VAR_INPUT
    eMode      : E_CX81x0_LED;
END_VAR
```

eMode: The way in which the LED lights up, see also Data type E_CX81x0_LED [▸ 44].

Return value

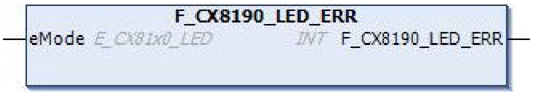
```
F_ CX8180_LED_WD       : INT;
```

F_ CX8180_LED_WD: -1, not implemented flash code, 0 OK

Requirements

Development environment	Target platform	Hardware	PLC libraries to include
TwinCAT v3.1 Build 4022.30	ARM	CXxxxx	Tc2_SystemCX

4.7 F_CX8190_LED_ERR



Since the CX8190 supports various protocols, the two LEDs WD and ERR on the CX8190 are not used by the firmware. This allows the user to create his own diagnosis messages. The LEDs can be used to indicate, for example, whether the CX8190 has received an IP address from the DHCP server or whether devices are exchanging data.

The F_CX8190_LED_ERR function controls the ERR LED on the CX8190. Various color and flashing modes can be used here. The possible LED colors are red and green.

VAR_INPUT

```
VAR_INPUT
    eMode      : E_CX81x0_LED;
END_VAR
```

eMode: The way in which the LED lights up, see also Data type E_CX81x0_LED [▶ 44].

Return value

```
F_ F_CX8190_LED_ERR     : INT;
```

F_CX8190_LED_ERR: -1, not implemented flash code, 0 OK

Requirements

Development environment	Target platform	Hardware	PLC libraries to include
TwinCAT v3.1 Build 4022.30	ARM	CXxxxx	Tc2_SystemCX

4.8 F_CX8190_LED_WD

F_CX8190_LED_WD
eMode
E_CX81x0_LED
INT
F_CX8190_LED_WD

Since the CX8190 supports various protocols, the two LEDs WD and ERR on the CX8190 are not used by the firmware. This allows the user to create his own diagnosis messages. The LEDs can be used to indicate, for example, whether the CX8190 has received an IP address from the DHCP server or whether devices are exchanging data.

The F_CX8190_LED_WD function controls the WD LED on the CX8190. Various color and flashing modes can be used here. The possible LED colors are red and green.

VAR_INPUT

```
VAR_INPUT
    eMode      : E_CX81x0_LED;
END_VAR
```

eMode: The way in which the LED lights up, see also Data type E_CX81x0_LED [▶ 44].

Return value

```
F_ CX8190_LED_WD      : INT;
```

F_ CX8190_LED_WD: -1, not implemented flash code, 0 OK

Requirements

Development environment	Target platform	Hardware	PLC libraries to include
TwinCAT v3.1 Build 4022.30	ARM	CXxxxx	Tc2_SystemCX

5 Data Types

5.1 [obsolete]

5.1.1 ST_CX_DeviceIdentification

```
TYPE ST_CxDeviceIdentification :
STRUCT
 strTargetType       : STRING(20);
 strHardwareModel    : STRING(10);
 strHardwareSerialNo : STRING(12);
 strHardwareVersion  : STRING(4);
 strHardwareDate     : STRING(10);
 strHardwareCPU      : STRING(10);
 strImageDevice      : STRING(20);
 strImageVersion     : STRING(10);
 strImageLevel       : STRING(10);
 strImageOsName      : STRING(20);
 strImageOsVersion   : STRING(8);
 strTwinCATVersion   : STRING(4);
 strTwinCATRevision  : STRING(4);
 strTwinCATBuild     : STRING(8);
 strTwinCATLevel     : STRING(20);
 strAmsNetId      : STRING(23);
END_STRUCT
END_TYPE
```

strTargetType: Type of the target system, e.g. 'CX1000-CE', ...

strHardwareModel: Hardware model, e.g. '1001'

strHardwareSerialNo: Hardware serial number, e.g. '123'

strHardwareVersion: Hardware version, e.g. '1.7'

strHardwareDate: Hardware production date, e.g. '18.8.06'

strHardwareCPU: Hardware CPU architecture, e.g. 'INTELx86', 'ARM', 'UNKNOWN' or '' (empty string)

strImageDevice: Software platform, e.g. 'CX1000', ...

strImageVersion: Version of the software platform, e.g. '2.15'

strImageVersion: Version of the software platform, e.g. 'HMI'

strImageOsVersion: Name of the operating system, e.g. 'Windows CE'

strImageOsVersion: Version of the operating system, e.g. '5.0'

strTwinCATVersion: TwinCAT version, e.g. for TwinCAT 2.10.1307: '2'

strTwinCATRevision: TwinCAT Reversion, e.g. for TwinCAT 2.10.1307: '10'

strTwinCATBuild: TwinCAT Build, e.g. for TwinCAT 2.10.1307: '1307'

strTwinCATLevel: Registered TwinCAT level, e.g. 'PLC', 'NC-PTP', 'NC-I', ...

strAmsNetId: TwinCAT AMS-NetID, e.g. '5.0.252.31.1.1'

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.0	CX (x86)	Tc2_SystemCX

5.1.2 ST_CxDeviceIdentificationEx

```
TYPE ST_CxDeviceIdentificationEx :
STRUCT
 strTargetType       : STRING(30);
 strHardwareModel    : STRING(16);
 strHardwareSerialNo : STRING(16);
 strHardwareVersion  : STRING(8);
 strHardwareDate     : STRING(12);
 strHardwareCPU      : STRING(20);
 strImageDevice      : STRING(48);
 strImageVersion     : STRING(32);
 strImageLevel       : STRING(32);
 strImageOsName      : STRING(48);
 strImageOsVersion   : STRING(8);
 strTwinCATVersion   : STRING(4);
 strTwinCATRevision  : STRING(4);
 strTwinCATBuild     : STRING(8);
 strTwinCATLevel     : STRING(20);
 strAmsNetId         : T_AMSNetId;
END_STRUCT
END_TYPE
```

strTargeType: Type of the target system, e.g. 'CX1000-CE', ...

strHardwareModel: Hardware model, e.g. '1001'

strHardwareSerialNo: Hardware serial number, e.g. '123'

strHardwareVersion: Hardware version, e.g. '1.7'

strHardwareDate: Hardware production date, e.g. '18.8.06'

strHardwareCPU: Hardware CPU architecture, e.g. 'INTELx86', 'ARM', 'UNKNOWN' or '' (empty string)

strImageDevice: Software platform, e.g. 'CX1000', ...

strImageVersion: Version of the software platform, e.g. '2.15'

strImageVersion: Version of the software platform, e.g. 'HMI'

strImageOsVersion: Name of the operating system, e.g. 'Windows CE'

strImageOsVersion: Version of the operating system, e.g. '5.0'

strTwinCATVersion: TwinCAT version, e.g. for TwinCAT 2.10.1307: '2'

strTwinCATRevision: TwinCAT Reversion, e.g. for TwinCAT 2.10.1307: '10'

strTwinCATBuild: TwinCAT Build, e.g. for TwinCAT 2.10.1307: '1307'

strTwinCATLevel: Registered TwinCAT level, e.g. 'PLC', 'NC-PTP', 'NC-I', ...

strAmsNetId: TwinCAT AMS-NetID, e.g. '5.0.252.31.1.1'

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.0	CX (x86)	Tc2_SystemCX

5.2 ST_CX_ProfilerStruct

```
TYPE ST_CX_ProfilerStruct:
STRUCT
 dwLastExecTime    : DWORD;
 dwMinExecTime     : DWORD;
 dwMaxExecTime     : DWORD;
 dwAverageExecTime : DWORD;
 dwMeasureCycle    : DWORD;
END_STRUCT
END_TYPE
```

dwLastExecTime: The most recently measured value for the execution time in [µs]

dwMinExecTime: The minimum execution time in [µs]

dwMaxExecTime: The maximum execution time in [µs]

dwAverageExecTime: The mean execution time for the last 100 measurements in [µs]

dwMeasureCycle: The number of measurements that have already been carried out

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.0	CX (x86)	Tc2_SystemCX

5.3 Data type E_CX81x0_LED

```
TYPE E_CX81x0_LED : (
    eCX81x0_LED_GREEN_OFF,
    eCX81x0_LED_GREEN_ON,
    eCX81x0_LED_GREEN_FLASHING_Quick,
    eCX81x0_LED_GREEN_FLASHING_200ms,
    eCX81x0_LED_GREEN_FLASHING_500ms,
    eCX81x0_LED_GREEN_FLASHING_Pulse,
    eCX81x0_LED_RED_OFF:=20,
    eCX81x0_LED_RED_ON,
    eCX81x0_LED_RED_FLASHING_Quick,
    eCX81x0_LED_RED_FLASHING_200ms,
    eCX81x0_LED_RED_FLASHING_500ms,
    eCX81x0_LED_RED_FLASHING_Pulse,
    eCX81x0_LED_GREEN_RED_OFF:=100,
    eCX81x0_LED_GREEN_RED_FLASHING_200ms,
    eCX81x0_LED_GREEN_RED_FLASHING_500ms
);
END_TYPE
```

6 Global constants

6.1 Library version

All libraries have a specific version. This version is inter alia shown in the PLC library repository too.
A global constant contains the library version information:

Global_Version

```
VAR_GLOBAL CONSTANT
    stLibVersion_Tc2_SystemCX : ST_LibVersion;
END_VAR
```

stLibVersion_Tc2_SystemCX: version information of the Tc2_SystemCX library (type: ST_LibVersion).

To compare the existing version to a required version the function F_CmpLibVersion (defined in Tc2_System library) is offered.

All other possibilities known from TwinCAT 2 to query a library version are obsolete!

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