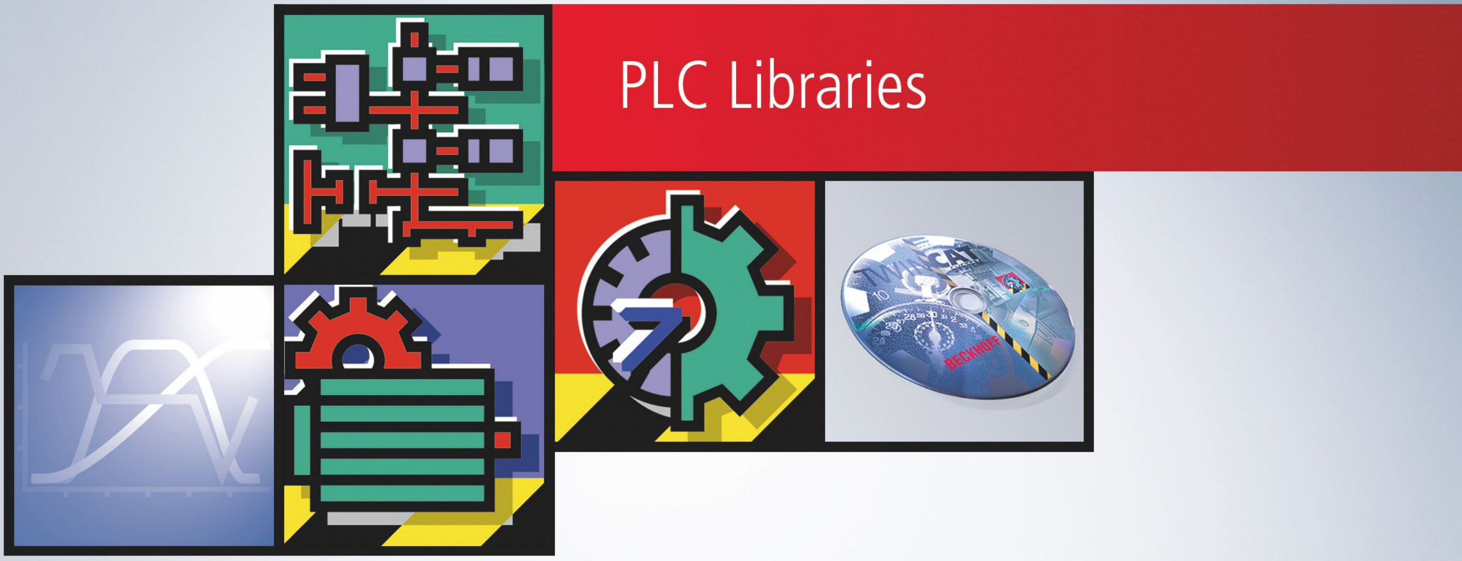
BECKHOFF New Automation Technology

Manual | EN

TX1200

TwinCAT 2 | PLC Library: TcSystemCX



2022-11-04 | Version: 1.1

Table of contents

1 Foreword

1.1 Notes on the documentation

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

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

Disclaimer

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

Trademarks

Patent Pending

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

Copyright

1.2 Safety instructions

Safety regulations

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

Exclusion of liability

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

Personnel qualification

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

Description of symbols

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

⚠ DANGER

Serious risk of injury!

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

⚠ WARNING

Risk of injury!

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

⚠ CAUTION

Personal injuries!

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

NOTE

Damage to the environment or devices

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

Tip or pointer

This symbol indicates information that contributes to better understanding.

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.

2 Overview

This library contains functions and function blocks which are using features of devices of the Embedded PC CX line .

Functions

Name	Description
F_CXSubTimeStamp [▶ 9]	calculates 64bit subtraction (time A [100ns] - time B [100ns]) as result in [µs], only with differences between 0 and 4294967295 us, see link
F_CXNaviSwitch [▶ 9]	The function converts the value of the CX1100 navigation switch into an enum.
F_GetVersionTcSystemCx [▶ 10]	The function returns library version info.

Function blocks

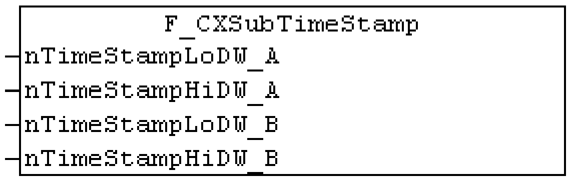
Name	Description
FB_CxGetDeviceIdentification [▶ 11]	Read the CX-Device identification
FB_CXProfiler [▶ 11]	runtime measurement of PLC code using the CPU-Counter
FB_CXSetTextDisplay [▶ 12]	Control of the two line display of the CX1100
FB_CXSimpleUps [▶ 13]	Control of the UPS CX1190-UPS (part name CX1100-0900, CX1100-0910, CX1100-0920)

Requirements

Component	Version
TwinCAT on the development PC	2.9 Build 959 or higher
CX1000-Windows CE-Image	1.75 or higher; FB_CxGetDeviceIdentification [▶ 11] since 2.15
CX1000-Windows XP-Image	1.14 or higher; FB_CxGetDeviceIdentification [▶ 11] higher than 1.33
CX1020-Windows CE-Image	2.04 or higher; FB_CxGetDeviceIdentification [▶ 11] since 2.15
CX1020-Windows XP-Image	1.30 or higher; FB_CxGetDeviceIdentification [▶ 11] higher than 1.33

3 Functions

3.1 F_CXSubTimeStamp



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

If the difference between time stamp A and time stamp B is negative or bigger than 4294967295us, then the maximum value 4294967295us is returned. This is 71 minutes, 34 seconds, 967 milli seconds and 295 micro seconds. In these cases the function UInt64Sub64() of the TcUtilities.lib can be used to calculate a 64-Bit subtraction with a 64-Bit result in [100ns].

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_VAR
```

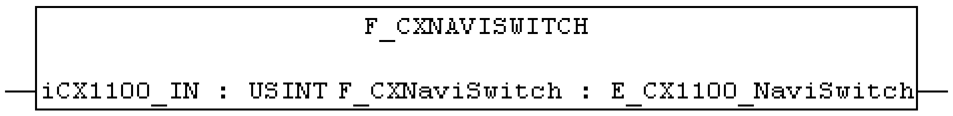
nTimeStampLoDW_A: Lower 32bit of time stamp A.

nTimeStampHiDW_A: Upper 32bit of time stamp A.

nTimeStampLoDW_B: Lower 32bit of time stamp B.

nTimeStampHiDW_B: Upper 32bit of time stamp B.

3.2 F_CXNaviSwitch



The function F_CXNaviSwitch converts the value of the CX1100-Navigation-Switch in to an Enum-value of the 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

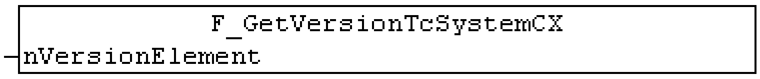
```
TYPE E_CX1100_NaviSwitch : (
    e_CX1100_NaviSwitch_IDLE        := 0,
    e_CX1100_NaviSwitch_MIDDLE          := 16,

    (* 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
```

Other values than defined in the enum (f.i. 11) are reported as "*** INVALID: value ***" in Online Mode (f.i. "*** INVALID: 11 ***"). The function F_CXNaviSwitch returns the invalid value (f.i. 11).

3.3 F_GetVersionTcSystemCX



The function returns library version info.

FUNCTION F_GetVersionTcSystemCX : UINT

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

nVersionElement : Version element:

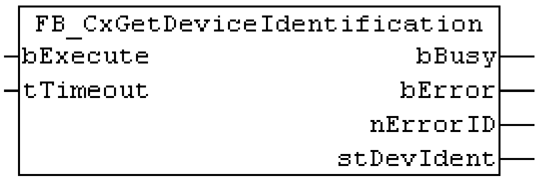
- 1 : major number;
- 2 : minor number;
- 3 : revision number;

Requirements

Development environment	Target plattform	PLC Libraries to include
TwinCAT v2.9.0	PC (i386)	TcSystemCX.Lib

4 Function Blocks

4.1 FB_CxGetDeviceIdentification



The functionblock FB_CxGetDeviceIdentification can be used to get the device identification of a CX-Device.

VAR_INPUT

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

bExecute: Command is executed with rising edge.

tTimeout: Timeout to cancel the function call.

VAR_OUTPUT

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

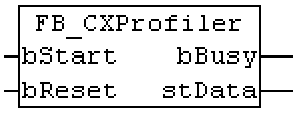
bBusy: Data are read from the CX-Device. Data are stored in stDevIdent if **bBusy = FALSE** and **bError = FALSE**.

bError: Gets TRUE, with any error.

nErrorID: Contains the error id if bErr is TRUE.

stDevIdent: Contains the read data, see ST_CxDeviceIdentification [▶ 16].

4.2 FB_CXProfiler



Please use this function block only under Windows CE. If your operating system is Windows XP or XPe. than please use the more accurate Profiler function block from the TcUtilities.Lib.

The Profiler function block can be used to allow the execution time of PLC code to be measured. 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 [▶ 16]. 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 [▶ 16] provides information about the number of measurements that have already been carried out. To measure the execution time for a specific segment of the PLC program the measurement must be started by a rising edge at the bStart input when the segment to be measured starts and stopped by a falling edge at the bStart input at the end of the segment. A rising edge at the RESET input and simultaneous rising edge at the START input will reset all the variables at the DATA output. The measurements in the stData structure that have already been determined then become invalid and are re-calculated when the function block is called again.

The times measured can differ from the true values, since a certain amount of time is needed just for the call of the GETCPUCOUNTER function block. This time depends on the computer and is included in the times that are found. Task interruptions i.e. by the NC are not detected and lead to higher values.

VAR_INPUT

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

bStart: A rising edge at this input starts the measurement of the execution time. A falling edge at this input stops the measurement, and causes the current, minimum, maximum and mean execution times to be recalculated. The variable stData.dwMeasureCycle is incremented at the same time.

bReset: A rising edge at this input will reset the variables at the stData output. A rising edge at this input and simultaneous rising edge at START input will reset the variables at the DATA output.

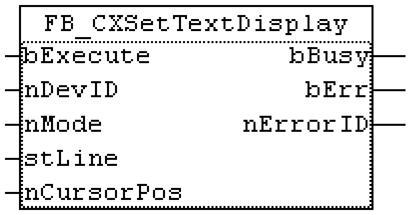
VAR_OUTPUT

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

bBusy: This output 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: Structure of type ST_CX_ProfilerStruct [▸ 16] with the measured times [in µs].

4.3 FB_CXSetTextDisplay



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

VAR_INPUT

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

bExecute: Command is executed with rising edge.

nDevID: Device ID of the CX1100-Device.

nMode: Modeswitch (see Enumeration)

stLine: String with 20 characters. This String is displayed with the appropriate command.

nCursorPos: Cursor position. The string is being written beginning from this position.

VAR_OUTPUT

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

bBusy: Command is being transported via ADS. bBusy remains TRUE, while no new command is accepted.

bErr: Gets TRUE, with any error.

nErrorID: Contains the error id if bErr is TRUE.

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_CX1000_DisplayNoAction: No Action.

e_CX1000_DisplayOn: Switches display on.

e_CX1000_DisplayOff: Switches display off.

e_CX1000_CursorOn: Switches cursor on.

e_CX1000_CursorOff: Switches cursor off.

e_CX1000_CursorBlinkOn: Switches blinking of cursor on.

e_CX1000_CursorBlinkOff: Switches blinking of cursor off.

e_CX1000_BackLightOn: Switches background light on.

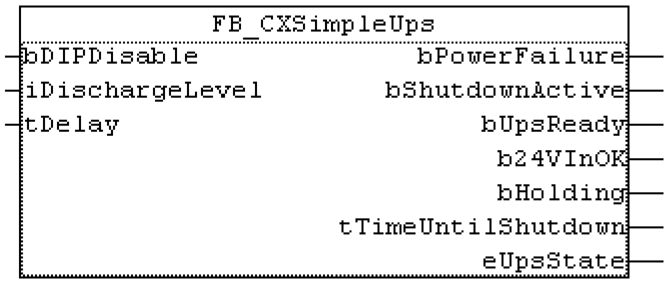
e_CX1000_BackLightOff: Switches background light off.

e_CX1000_ClearDisplay: Clears display.

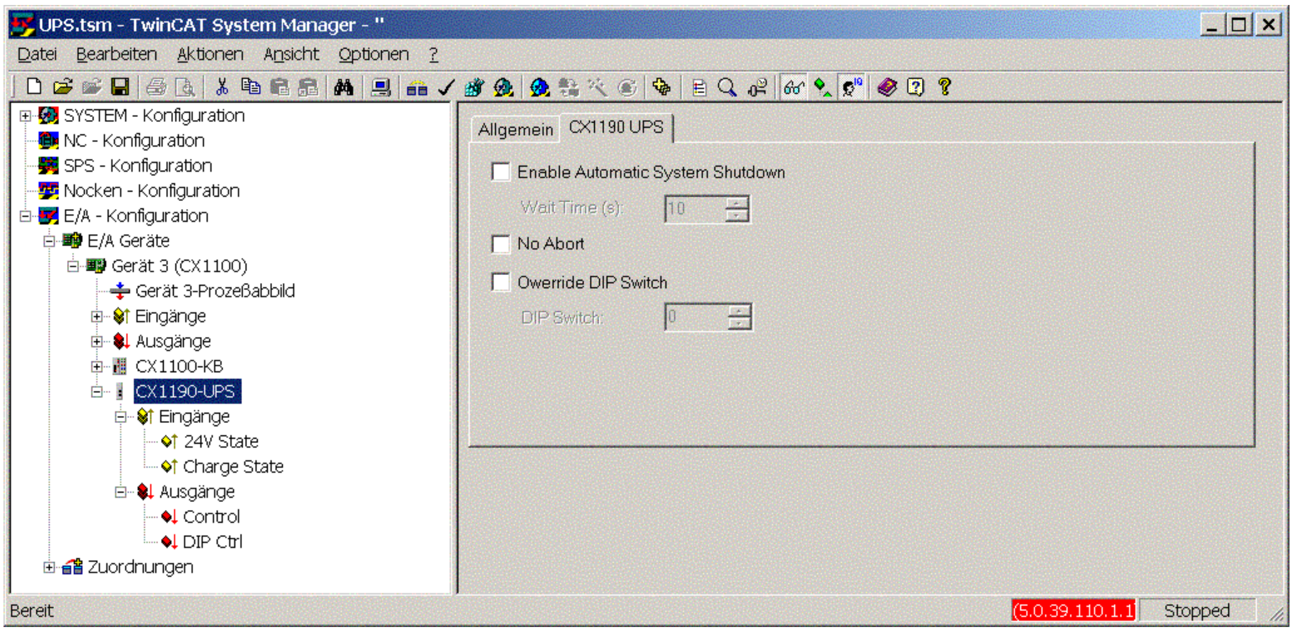
e_CX1000_WriteLine1: Write line 1.

e_CX1000_WriteLine2: Write line 2.

4.4 FB_CXSimpleUps



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



Enable Automatic System Shutdown: ☐
Wait Time (s): 10
No Abort: ☐
Owerride DIP Switch: ☐
DIP Switch: 0

VAR_INPUT

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

bDIPDisable: If TRUE, then the orientation of the '1-2-3..8-9-0' switch on the UPS is ignored, in this case the iDischargeLevel is used instead.

iDischargeLevel: Discharge Switch Off Level: 0 = 100% (Maximum Discharge), 9 = 90%, 8 = 80%, ..., 2 = 20%, 1 = 10% (Minimum Discharge).

tDelay: Holding time, time before the Shutdown is executed. This time is used to overcome short power outages (up to 10s) without shutdown. Once the holding time is elapsed, the FB finishes the holding period and waits internally for additional 2.5s. If the power has returned by then, the FB continues with normal run mode, otherwise the powerfail shutdown is executed. If the power returns during or after the shutdown, then the CX reboots after discharging and recharging the UPS.

VAR_OUTPUT

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

bPowerFailure: Gets TRUE, if a power failure of the power supply is detected, gets FALSE, if the power supply is restored.

bShutdownActive: Gets TRUE, if a Stop or Shutdown is being executed.

bUpsReady: Gets TRUE, if the UPS supplies the voltage.

b24VInOK: Gets TRUE, if the power supply supplies the UPS with 24V.

bHolding: Gets TRUE, if a power failure of the power supply is detected, and the holding time has not yet elapsed.

tTimeUntilShotdown: Shows the time until the system shuts down after a power fault.

eUpsState: Shows the status of the UPS [UNDEF | CHARGING | CHARGED | DISCHARGE | DISCHARGE_RESTART | OUTPUT_OFF | OVERLOAD].

VAR_CONFIG

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

Ii24VState: Needs to be linked to input '24V State', see picture above.

IiChargeState: Needs to be linked to input 'Charge State', see picture above.

QiControl: Needs to be linked to output 'Contol', see picture above.

QiDIPControl: Needs to be linked to output 'DIP Ctrl', see picture above.

5 Data Types

5.1 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, f.i. 'CX1000-CE', ...

strHardwareModel: Hardware-Model, f.i. '1001'.

strHardwareSerialNo: Hardware-Serialnumber, f.i. '123'.

strHardwareVersion: Hardware-Version, f.i. '1.7'.

strHardwareDate: Hardware-Production-Date, f.i. '18.8.06'.

strHardwareCPU: Hardware-CPU-Architecture, f.i. 'INTELx86', 'ARM', 'UNKNOWN' or '' (empty string).

strImageDevice: Software-Platform, f.i. 'CX1000', ...

strImageVersion: Version of Software-Platform, f.i. '2.15'.

strImageLevel: Level of Software-Platform, f.i. 'HMI'.

strImageOsName: Name of Operatingsystem, f.i. 'Windows CE'.

strImageOsVersion: Version of Operatingsystem, f.i. '5.0'.

strTwinCATVersion: TwinCAT Version, f.i. for TwinCAT 2.10.1307: '2'.

strTwinCATRevision: TwinCAT Revision, f.i. for TwinCAT 2.10.1307: '10'.

strTwinCATBuild: TwinCAT Build, f.i. for TwinCAT 2.10.1307: '1307'.

strTwinCATLevel: Registered TwinCAT Level, f.i. 'PLC', 'NC-PTP', 'NC-I', ...

strAmsNetId: TwinCAT AMS-NetID, f.i. '5.0.252.31.1.1'.

5.2 ST_CX_ProfilerStruct

```
TYPE ST_CX1000_ProfilerStruct:
STRUCT
   dwLastExecTime      :DWORD;
   dwMinExecTime       :DWORD;
```

```
    dwMaxExecTime       :DWORD;
    dwAverageExecTime   :DWORD;
    dwMeasureCycle      :DWORD;
END_STRUCT
END_TYPE
```

dwLastExecTime: Last measured value of execution time in [µs].

dwMinExecTime: Minimum execution time in [µs].

dwMaxExecTime: Maximum execution time in [µs].

dwAverageExecTime: Average execution time of last 100 measurements in [µs].

dwMeasureCycle: Amount of already measured cycles.

More Information:
www.beckhoff.com/tx1200

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