sysWORXX CTR-700

Quick Start Guide

Project:	sysWORXX CTR-700
Project Number:	131606
Document Revision:	L-2101e-05
Author:	Ferenc Reményi;Christian Schuster;Andreas Dinter
File Name:	L-2101e_05_10 Quick Start Guide
Print Date, Time:	19.04.2018, 14:37
Customer:	-
Classification:	public

Release

Version	Created	Checked	Released
01	F. Reményi – 07.11.2017		SYS TEC:- Customer:-

Revision History

Version	Date	Author	Changes
01	07.11.2017	F. Reményi	
02	27.11.2017	C. Schuster	Overview Configuration Application Development Process Image
03	05.12.2017	A. Dinter	Start/Stop OPC UA Server Setup OpenPCS Project for use with OPC UA Server
04	06.02.2018	A. Dinter	I/O Driver / C/C++ Development
05	29.03.2018	A. Dinter	Update service names for OpenPCS

Contents

List of Tables

List of Figures

1 Introduction

The document contains an overview about terminals of the sysWORXX CTR-700 and describes the first steps in software.

Figure 1: sysWORXX CTR-700

2 Overview

Table 1: Overview of relevant manuals

Information about…	**In which manual?**
Basic information about the CTR-700 (connections, configuration, administration, process image)	In this manual
Basics about the *OpenPCS* IEC 61131 programming system	Brief instructions for the programming system (Entry "*OpenPCS Dokumentation*" in the *OpenPCS* program group of the start menu) (Manual no.: L-1005)
Complete description about the *OpenPCS* IEC 61131 programming system, basics of PLC programming according to IEC 61131-3	Online help about the *OpenPCS* programming system
Command overview and description of standard function blocks according to IEC 61131-3	Online help about the *OpenPCS* programming system
SYS TEC extension for IEC 61131-3: - String functions - UDP function blocks - SIO function blocks - FB for RTC, Counter, EEPROM, PWM/PTO	User Manual *"SYS TEC-specific extensions for OpenPCS / IEC 61131-3"* (Manual no.: L-1054)
CANopen extension for IEC 61131-3 (network variables, *CANopen* function blocks)	User Manual *"CANopen extension for IEC 61131-3"* (Manual no.: L-1008)
Textbook about PLC programming according to IEC 61131-3	IEC 61131-3: Programming Industrial Automation Systems John/Tiegelkamp Springer-Verlag ISBN: 3-540-67752-6 (a short version is available as PDF on the *OpenPCS* installation CD)

3 Description Hardware

3.1 Pin assignment

Figure 2: Overview Terminals

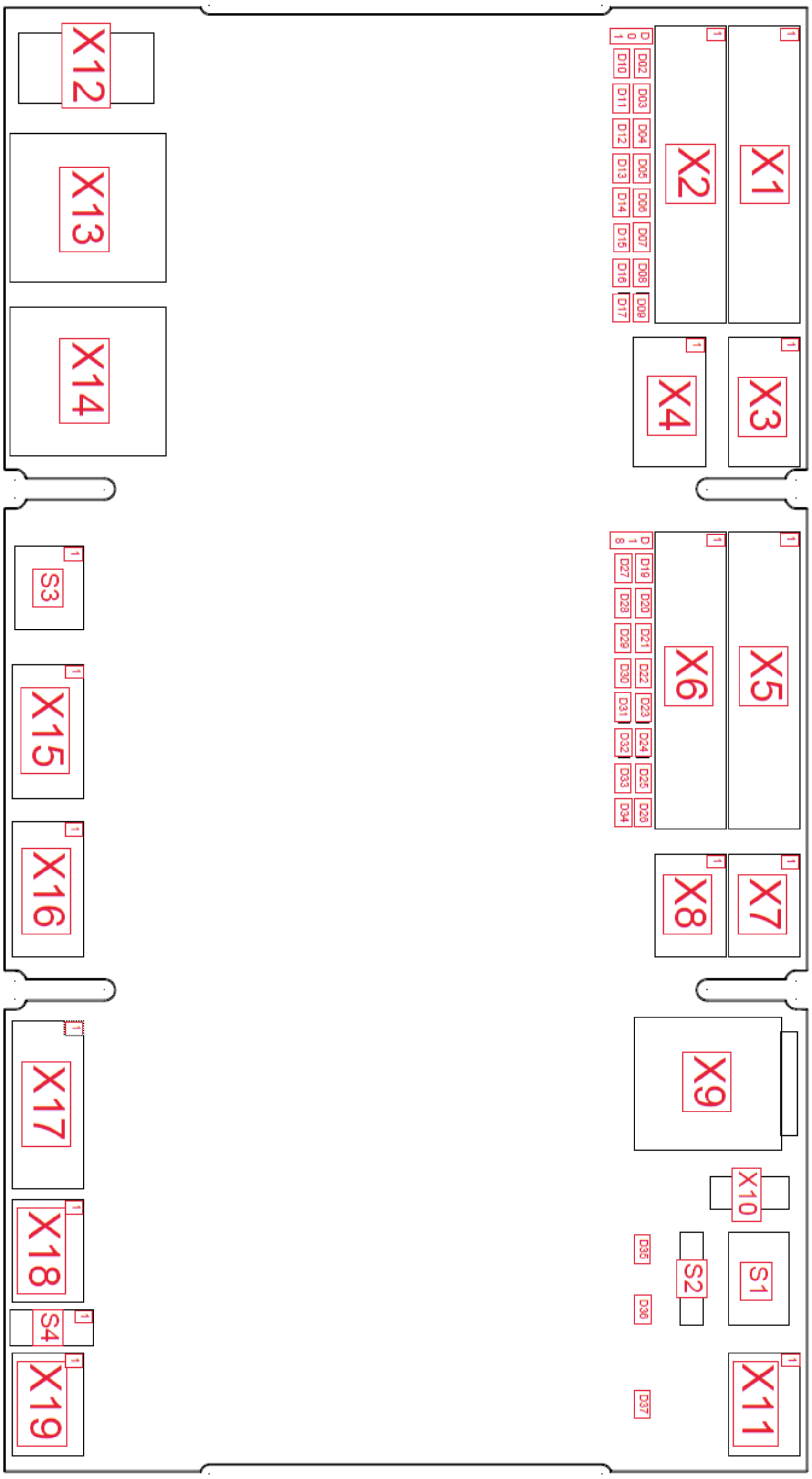


Table 2: Description Terminals

Terminal	Port	Signal name	Interface
X1	1	GND	digital outputs 0 … 15 (24V)
	2 … 9	DO0 … DO7	
X2	1	24V	
	2 … 9	DO8 … DO15*	
X3	1	COM	relay 0 (230V)
	2	NO	
	3	NC	
X4	1	COM	relay 1 (230V)
	2	NO	
	3	NC	
X5	1	GND	digital inputs 0 … 15 (24V)
	2 … 9	DI0 … DI7	
X6	1	24V	
	2 … 9	DI8 … DI15	
X7	1, 2	AIN 0, AIN 1	analog inputs 0 … 3 (0 … 10V/0 … 20mA)
	3	AGND	
X8	1, 2	AIN 2, AIN 3	
	3	AGND	
X9	-	-	µSD-card-holder
X10	-	-	µUSB (console)
X11	1	PE	power (24V)
	2	GND	
	3	24V	
X12	-	-	USB-Host
X13	-	-	ethernet 0
X14	-	-	ethernet 1
X15	1	RX, D0	serial interface 0 (RS232/RS485-MOD-BUS)
	2	D1	
	3	TX	
	4	GND	
X16	1	RX, D0	serial interface 1 (RS232/RS485-MOD-BUS)
	2	D1	
	3	TX	
	4	GND	
X17	1	RX, D0	serial interface 2 (RS232/RS485-MOD-BUS)
	2	CTS, D1	
	3	TX	
	4	RTS	
	5	GND	
X18	1	HIGH	CAN 0
	2	LOW	
	3	GND	
X19	1	HIGH	CAN 1
	2	LOW	
	3	GND	

* DO10 and DO11 are used also as boot configuration pins. Depending on the configured boot mode (SD-Card/EMMC) this will cause one of the outputs to activate for about one second when booting the CTR-700. This issue will be resolved in an upcoming revision.

Table 3: Description Switches

Switch	Port	Feature	0	1
S1	-	RESET	-	-
S2	-	RUN	OFF	ON
S3	1	termination serial interface 2 (RS485-MOD-BUS)	OFF	ON
	2	termination serial interface 1 (RS485-MOD-BUS)	OFF	ON
	3	termination serial interface 0 (RS485-MOD-BUS)	OFF	ON
	4	configuration	OFF	ON
	5	boot	-	-
	6	boot mode (SD-Karte/EMMC)	SD	EMMC
S4	1	termination CAN 0	OFF	ON
	2	termination CAN 1	OFF	ON

Table 4: Description LEDs

Led	Colour	Feature
D01	green	status of power supply digital outputs
D02 … D17	yellow	status of digital outputs
D18	green	status power supply digital inputs
D19 … D34	yellow	status digital inputs
D35	green	status of the system (RUN)
D36	red	status of the system (STOP)
D37	green	power supply device

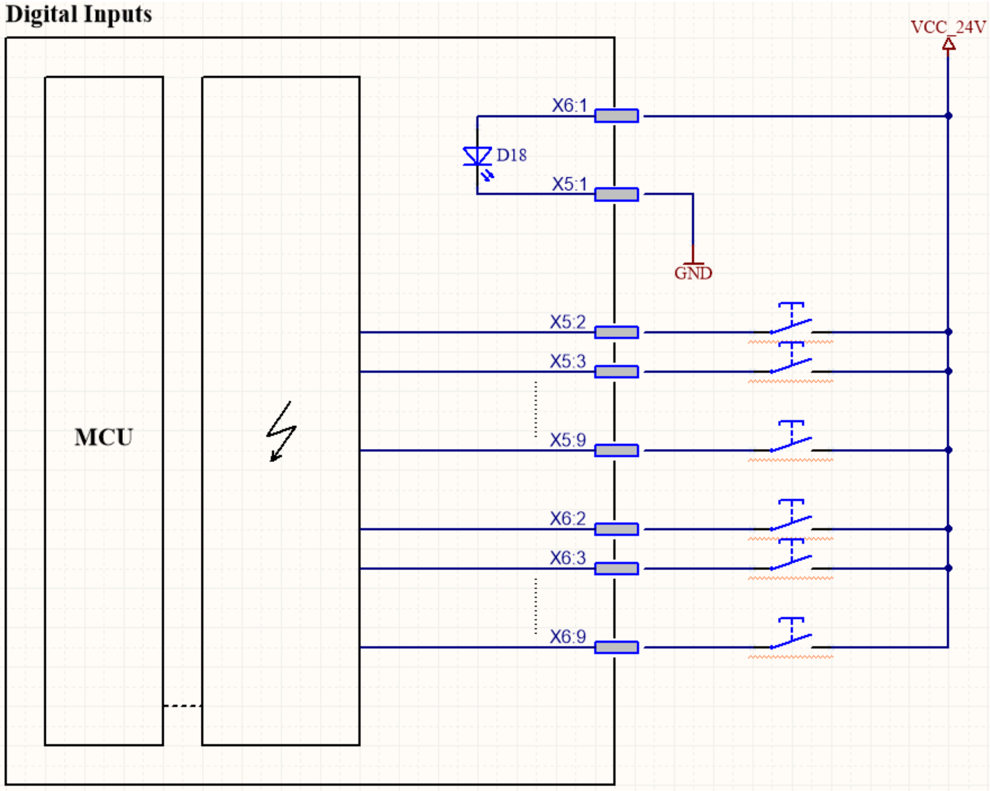
3.2 Digital inputs DI0 ... DI15 (24VDC)

The CTR-700 features 16 digital inputs (DI0 ... DI15). The inputs are galvanically isolated in groups of eight inputs. Each four inputs have the same supply potential (DI0 … 7, DI8 … 15). The inputs are highly active with the following selector shaft:

- Input voltage > 15 VDC: is shown as ‘1’ in the process image
- Input voltage < 5 VDC: is shown as ‘0’ in the process image

Digital inputs DI0 ... DI15 have the internal structure as shown in Figure 3.

Figure 3: Setup of digital inputs DI0 … DI15



The digital inputs in a PLC program are accessible via the process image (see in Table 5 section 6.1).

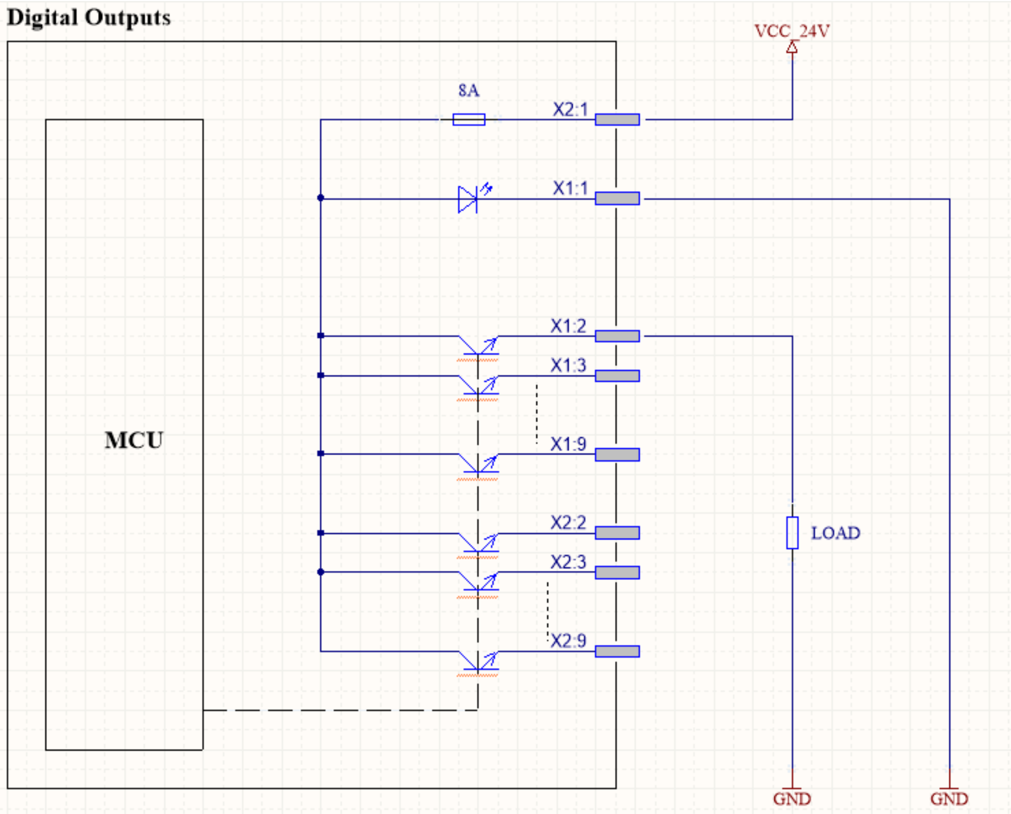
3.3 Digital outputs DO0 ... DO15 (24VDC / 0.5A, short-circuit-proof)

The CTR-700 features 16 digital transistor outputs (DO0 ... DO15). The outputs each connect the supply voltage Vcc of the appliance (switching positively). The maximum load current for each 24V output is 0.5A for ohmic, inductive or capacitive load. The outputs are short-circuit-proof and galvanically isolated from the CPU unit. The performance drivers used are protected against excess voltage, reverse polarity and excess temperature. The transistor outputs are accessed high-actively:

- ‘1’ in process image: output transistor active, appliance connected with Vcc
- ‘0’ in process image: output transistor inactive, appliance disconnected from Vcc

The digital transistor outputs DO0 ... DO15 have the internal structure as shown in Figure 4 .

Figure 4: Setup of digital outputs DO0 ... DO15



The digital outputs in a PLC program are accessible via the process image (see in Table 5 section 6.1).

4 Configuration and Administration of the CTR-700

The Operating System of the CTR-700 is a Debian GNU/Linux installed on the included SD card. This means you can install software via apt/apt-get utilities.

There are two options to access the CTR-700 via command shell:

- SSH via eth0, which is by default configured for DHCP
- Serial connection via USB-Service plug (On board USB-FTTI chip)
    - Baudrate: 115200 / 8N1

The default login credentials are as follows:

User:	root
Password:	root

4.1 System requirements and necessary software tools

The administration of the CTR-700 requires any Windows or Linux computer that has available an Ethernet or USB interface. These allow a connection to administer the CTR-700 via a Linux command line-interface.

All examples referred to in this manual are based on an administration of the CTR-700 using a Windows computer. Procedures using a Linux computer would be analogous.

To administrate the CTR-700 the following software tools are necessary:

Terminal program

A Terminal program allows the communication with the **command shell** of the CTR-700 via **the integrated USB-to-UART bridge (USB service console) of the** CTR-700. This is required for the Ethernet configuration of the CTR-700 as described in section 4.4. After completing the Ethernet configuration, all further commands can either be entered in the Terminal program or alternatively in a SSH client (see below).

A suitable Terminal program would be *"TeraTerm",* which is available as Open Source Software (*BSD* License). The project page is located at: http://ttssh2.osdn.jp/.

SSH

SSH allows the encrypted communication with **command shell** of the CTR-700 via **Ethernet**. Using SSH requires a completed Ethernet configuration of the CTR-700 according to section 4.4. As alternative solution to SSH, all commands can be used via a Terminal program.

Suitable as SSH client would be *"PuTTY"* or *"TeraTerm",* which can also be used as Terminal program (see above). "*PuTTY*" is licensed under *MIT*-License and can be downloaded at: https://www.chiark.greenend.org.uk/~sgtatham/putty/.

SFTP client

An SFTP client allows for file transfer between the CTR-700 and the computer. This allows for example **editing configuration files** by transferring those from the CTR-700 onto the computer where they can be edited and get transferred back to the CTR-700. Downloading files onto the CTR-700 is also necessary to **update the PLC firmware**. (Advice: The update of *PLC firmware* is not identical with the update of the *PLC user program*. The PLC program is directly transferred to the module from the *OpenPCS* programming environment. No additional software is needed for that.)

Suitable as SFTP client would be *"WinSCP"* which is available as Open Source Software (GNU GPL). It can be downloaded from the project page: http://winscp.net).

4.2 Automatic execution of services on Linux start-up

There are two options to start scripts or software automatically on startup/reset of the CTR-700.

4.2.1 Extend shell script in /etc/rc.local

The start script */etc/rc.local* will be executed automatically at startup of the system. This file can be altered by the user to execute additional shell commands. One has to keep in mind to not block the execution of the script for a long time or start long running commands in background.

4.2.2 Add a systemd service

System service files have to be added in */etc/systemd/system/<YOUR_ SERVICE >.service*. The service-file has to contain at least the following options:

```
[Unit]
Description=<YOUR_SERVICE_DESCRIPTION>
[Service]
ExecStart=/usr/bin/YOUR_SERVICE
[Install]
WantedBy=multi-user.target
```

Description is the name for the service and *ExecStart* is the path to the executable file or script.

You can add it to the autostart with the following command:

```
> systemctl enable YOUR_SERVICE
```

You can also remove it with the following command:

```
> systemctl disable YOUR_SERVICE
```

As more in-depth examples, one could look up the service files of *OpenPCS* or *Node-RED*. The official documentation can be found in the provided man-pages or on the project site of *systemd*.
Important man-pages regarding system services:

```
> man systemd
> man systemd.unit
> man systemd.service
```

Project homepage: https://www.freedesktop.org/wiki/Software/systemd/

4.3 OpenPCS, OPC UA and Node-RED Services

The usage of *OpenPCS, OPC UA* or *Node-RED* is only possible, if the corresponding service is running on the CTR-700. There are two ways to start *OpenPCS, OPC UA Server* or *Node-RED* on the CTR-700:

1. The following commands are used to manually start or stop *OpenPCS* services on the CTR-700:
   ```
   > systemctl start openpcs-z5
   > systemctl stop openpcs-z5
   ```

2. There is also the possibility to start the *OpenPCS* services automatically on PowerOn or Reset. These are the same commands as mentioned in Section 4.2.2:
   ```
   > systemctl enable openpcs-z5
   ```
   To disable the automatic start, the following command is used:
   ```
   > systemctl disable openpcs-z5
   ```

There are 2 variants of the *OpenPCS* runtime service `openpcs-z5` and `openpcs-z4`. You can choose one of them to determine the way of communicating with *OpenPCS. (*`z4` for CAN bus, `z5` for Ethernet / UDP*).*

To do the same with the *OPC UA Server* or *Node-RED* just replace *openpcs* with *opcua-server or node-red* respectively.

If the software is running on the CTR-700, you can connect with *OpenPCS* (for more Information see Table 1). Use some *OPC-UA client* to connect to the *OPC-UA Server*. To use *Node-RED*, you have to connect via a browser (e.g. *Firefox*). You have to use the assigned IP and the port 1880.

4.4 Network configuration

The CTR-700 has two network interfaces called **eth0** and **eth1**. The configuration file is */etc/network/interfaces*. The examples below use **ethX** as placeholder. Substitute **ethX** by the targeted network interface.
By default, only **eth0** is configured to use DHCP. The interface **eth1** has no configuration at all.

4.4.1 DHCP configuration

Add the configuration options listed below to the configuration file, to change the network interface to DHCP.

```
allow-hotplug ethX
iface ethX inet dhcp
```

4.4.2 Static IP address configuration

Add the configuration options listed below to the configuration file, to change the network interface to static. Use the proper configuration for your network infrastructure.

```
allow-hotplug ethX
iface ethX inet static
      address     192.168.0.100
      netmask     255.255.255.0
      network     192.168.0.0
      broadcast   192.168.0.255
      gateway     192.168.0.1
```

5 Application Development for the CTR-700

5.1 Running C# Applications via Mono

C# applications can be started like regular Linux applications by being called by their filename. An explicit call from mono is not necessary. C# .NET applications can be transferred via SFTP (see 4.1) to the CRT-700. Also, the application needs the permission added to be executable:

```
> chmod +x YOUR_APP_NAME
```

The application can then be started from its directory:

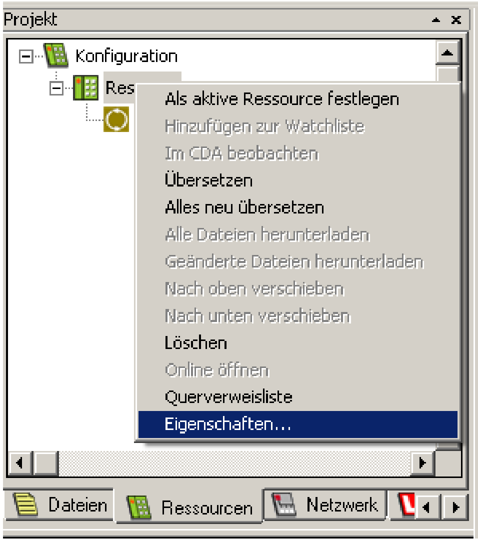
```
> root@ctr700:~/mono# ./CTR700_Hello_Word.exe
  Hello CTR-700, hello World!
```

5.2 Setup a OpenPCS Project for OPC UA Server

To allow the *OPC UA Server* to provide the Variables of an *OpenPCS* program one has to do some additional setup steps.

Create a new OpenPCS project with the Type *SYSTEC PLC (without Visualization)*, give it a name and add code and variables as needed. In the *Project* panel switch to the tab *Resources*, right click on the *Resource* element in the tree view and select preferences:

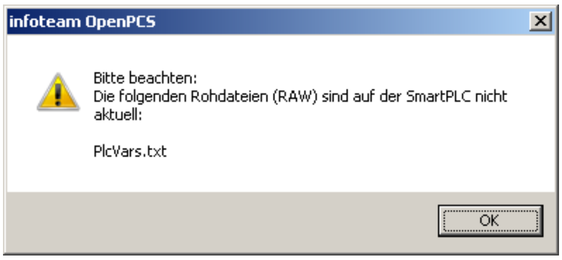
Figure 5 - OpenPCS Ressource Panel



Select *SYSTEC – sysWORXX CTR-700/Z5* as your *Hardware Module* and choose the proper *Network Connection* to the target device. After this compile your project. Switch the *Project* panel to the *File* view. Now you should be able to see a new file called *PlcVars.txt.* This file provides information for the *OPC UA Server* to build up the variable tree. Therefore one has to download this file to the target. To do this, right click on the file entry *PlcVars.txt* and select *Add to current resource.*
Ensure the services for *OpenPCS* and *OPC UA Server* are running on the target device (see 4.3 for the necessary steps).
Now go online and confirm downloading the *Resource* if necessary.

In case you get the following warning, switch to the *Resources* tab, right click on the *Files* element in the Resource tree and select “Download all files”. This will make sure that the variables information of *OpenPCS* and the *OPC UA Server* are synchronized.

Figure 6 - OpenPCS warning for unsynchronized data



After all files are downloaded successfully, start the *PLC* application.

Now you are able to connect with an arbitrary *OPC-UA Client* to the *Server* and see the variable tree of the *OpenPCS* program. The server is able to read and write to the variables of the *OpenPCS* program.

Figure 7 - OPC UA Client Settings

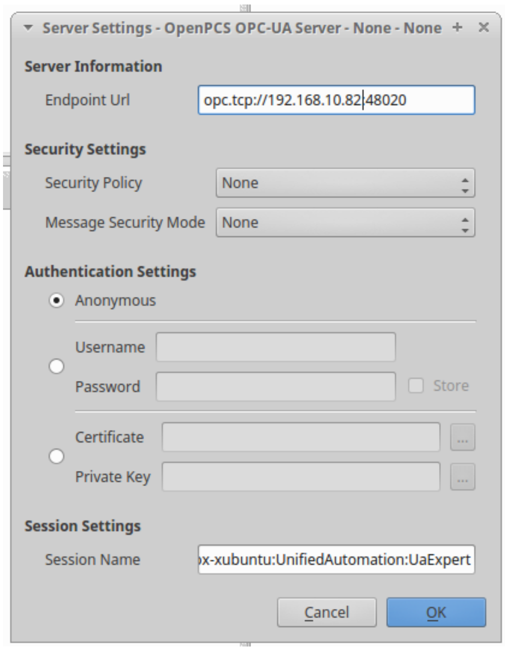
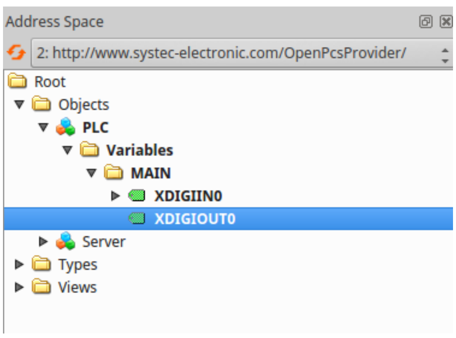


Figure 8 - OPC UA client variables tree



5.3 I/O Driver and C/C++ Development

The user space I/O Driver of the *sysWORXX CTR-700* is installed as a standard *Debian* package called *libctr700drv*. This can be used with several programming languages including *C, Java, Java-Script* or *C#*.

The driver is installed to the following files/directories:

/usr/lib/arm-linux-gnueabihf/libctr700drv.so	Actual driver library
/usr/include/ctr700drv/ctr700drv.h	Header-file for the driver library
/usr/share/libctr700drv	This directory contains examples for using the driver from C, Java, Java-Script or C#

6 OpenPCS Process Image of the CTR-700

6.1 Local in- and Outputs

Compared to other SYS TEC compact control systems, the CTR-700 obtains a process image with identical addresses. All in- and outputs listed in Table 5 are supported by the CTR-700.

Table 5: Assignment of in- and outputs to the process image of the CTR-700

I/O of the CTR-700	Address and Data type in the Process Image	
DI0 ... DI7	**%IB0.0** %IX0.0 … %IX0.7	as Byte with DI0 … DI7 as single Bit for each input
DI8 ... DI15	**%IB1.0** %IX1.0 … %IX1.7	as Byte with DI8 … DI15 as single Bit for each input
AI0	**%IW8.0**	15Bit + sign (0 … +32767)
AI1	**%IW10.0**	15Bit + sign (0 … +32767)
AI2	**%IW12.0**	15Bit + sign (0 … +32767)
AI3	**%IW14.0**	15Bit + sign (0 … +32767)
C0(1)	**%ID40.0** counter input: DI24 (%IX3.0), direction: DI21 (%IX2.5)	31Bit + sign (-2^{31} - 2^{31} -1)
C1(1)	**%ID44.0** counter input: DI25 (%IX3.1), direction: DI22 (%IX2.6)	31Bit + sign (-2^{31} - 2^{31} -1)
CPU Temperature Sensor	**%ID72.0**	31Bit + sign as 1/10000 °C
System Temperature Sensor	**%ID76.0**	31Bit + sign as 1/10000 °C
DO0 ... DO7	**%QB0.0** %QX0.0 … %QX0.7	as Byte with DO0 … DO7 as single Bit for each output
DO8 … DO15	**%QB1.0** %QX1.0 … %QX1.7	as Byte with DO8 … DO15 as single Bit for each output
REL0 and REL1 (corresponds to DO16 … DO17)	**%QB2.0** %QX2.0 … %QX2.1	as Byte with REL0 and REL1 as single Bit for each Relay

(1) Counters have not yet been implemented. Full function will be added in a future update.

Advice: The CTR-700 works with Little-Endian format (“Intel-Notation). Consequently, and on the contrary to controls using Big-Endian (“Motorola-Notation), it is **possible** to sum up several BYTE variables of the process image to one WORD or DWORD and to access Bits above Bit7. The following example shows issue described:

```
bInByte0  AT %IB0.0 : BYTE;
bInByte1  AT %IB1.0 : BYTE;
wInWord0  AT %IW0.0 : WORD;

wInWord0.0 == bInByte0.0     due to Little-Endian:  wInWord0.0 <> bInByte1.0
wInWord0.8 == bInByte1.0     due to Little-Endian:  wInWord0.8 <> bInByte0.0
```

In- and outputs of the CTR-700 are not negated in the process image. Hence, the H-level at one input leads to value "1" at the corresponding address in the process image. Contrariwise, value "1" in the process image leads to an H-level at the appropriate output.

6.2 Network variables for CAN1

Contrary to interface CAN0, interface CAN1 of the CTR-700 is designed as static object dictionary. Thus, at interface CAN1 the CTR-700 acts as a CANopen I/O device. All static network variables for CAN1 are accessible via the marker section of the process image.

On the contrary to interface CAN0, interface CAN1 is provided as static object dictionary. This means that the amount of network variables (communication objects) and the amount of PDOs available are both strongly specified. During runtime, the configuration of PDOs is modifiable. This implies that communication parameters used (CAN Identifier, etc.) and the allocation of network variables to each Byte of a CAN telegram (mapping), can be set and modified by the user. Thus, only the amount of objects (amount of network variables and PDOs) is strongly specified in the static object dictionary. Consequently, application and characteristics of objects can be modified during runtime. For this reason, at interface CAN1 the CTR-700 acts as a CANopen I/O device.

All network variables of the PLC program are available through the marker section of the process image. Therefore, 252 Bytes are usable as input variables and also 252 Bytes as output variables. To enable any data exchange with other CANopen I/O devices, the section of static network variables is mapped to different data types in the object dictionary (BYTE, SINT, WORD, INT, DWORD, DINT). Variables of the different data types are located within the same memory area which means that all variables represent the same physical storage location. Hence, a WORD variable interferes with 2 BYTE variables, a DWORD variable with 2 WORD or 4 BYTE variables. Figure 9 shows the positioning of network variables for CAN1 within the marker section.

Figure 9: Positioning of network variables for CAN1 within the marker section

CAN1 Input Variables

	CAN1 IN0	CAN1 IN1	CAN1 IN2	CAN1 IN3	CAN1 IN4	CAN1 IN5	CAN1 IN6	CAN1 IN7	...	CAN1 IN244	CAN1 IN245	CAN1 IN246	CAN1 IN247	CAN1 IN248	CAN1 IN249	CAN1 IN250	CAN1 IN251
BYTE / SINT, USINT	%MB 0.0 (Byte0)	%MB 1.0 (Byte1)	%MB 2.0 (Byte2)	%MB 3.0 (Byte3)	%MB 4.0 (Byte4)	%MB 5.0 (Byte5)	%MB 6.0 (Byte6)	%MB 7.0 (Byte7)		%MB 244.0 (Byte244)	%MB 245.0 (Byte245)	%MB 246.0 (Byte246)	%MB 247.0 (Byte247)	%MB 248.0 (Byte248)	%MB 249.0 (Byte249)	%MB 250.0 (Byte250)	%MB 251.0 (Byte251)
WORD / INT, UINT	%MW 0.0 (Word0)		%MW 2.0 (Word1)		%MW 4.0 (Word2)		%MW 6.0 (Word3)		...	%MW 244.0 (Word122)		%MW 246.0 (Word123)		%MW 248.0 (Word124)		%MW 250.0 (Word125)	
DWORD / DINT, UDINT	%MD 0.0 (Dw ord0)				%MD 4.0 (Dw ord1)					%MD 244.0 (Dw ord61)				%MD 248.0 (Dw ord62)			

CAN1 Output Variables

	CAN1 OUT0	CAN1 OUT1	CAN1 OUT2	CAN1 OUT3	CAN1 OUT4	CAN1 OUT5	CAN1 OUT6	CAN1 OUT7	...	CAN1 OUT244	CAN1 OUT245	CAN1 OUT246	CAN1 OUT247	CAN1 OUT248	CAN1 OUT249	CAN1 OUT250	CAN1 OUT251
BYTE / SINT, USINT	%MB 256.0 (Byte0)	%MB 257.0 (Byte1)	%MB 258.0 (Byte2)	%MB 259.0 (Byte3)	%MB 260.0 (Byte4)	%MB 261.0 (Byte5)	%MB 262.0 (Byte6)	%MB 263.0 (Byte7)		%MB 500.0 (Byte244)	%MB 501.0 (Byte245)	%MB 502.0 (Byte246)	%MB 503.0 (Byte247)	%MB 504.0 (Byte248)	%MB 505.0 (Byte249)	%MB 506.0 (Byte250)	%MB 507.0 (Byte251)
WORD / INT, UINT	%MW 256.0 (Word0)		%MW 258.0 (Word1)		%MW 260.0 (Word2)		%MW 262.0 (Word3)		...	%MW 500.0 (Word122)		%MW 502.0 (Word123)		%MW 504.0 (Word124)		%MW 506.0 (Word125)	
DWORD / DINT, UDINT	%MD 265.0 (Dw ord0)				%MD 260.0 (Dw ord1)					%MD 500.0 (Dw ord61)				%MD 504.0 (Dw ord62)			

Table 6 shows the representation of network variables through appropriate inputs in the object dictionary of interface CAN1.

Table 6: Representation of network variables for CAN1 by entries in the object dictionary

OD section	OD variable / EDS input	Data type CANopen	Data type IEC 61131-3
Inputs (inputs for the CTR-700*)*			
Index 2000H Sub 1 … 252	CAN1InByte0 … CAN1InByte251	Unsigned8	BYTE, USINT
Index 2001H Sub 1 … 252	CAN1InSInt0 … CAN1InSInt251	Integer8	SINT
Index 2010H Sub 1 … 126	CAN1InWord0 … CAN1InWord125	Unsigned16	WORD, UINT
Index 2011H Sub 1 … 126	CAN1InInt0 … CAN1InInt125	Integer16	INT
Index 2020H Sub 1 … 63	CAN1InDword0 … CAN1InDword62	Unsigned32	DWORD, UDINT
Index 2021H Sub 1 … 63	CAN1InDInt0 … CAN1InDInt62	Integer32	DINT
Outputs (outputs for the CTR-700*)*			
Index 2030H Sub 1 … 252	CAN1OutByte0 … CAN1OutByte251	Unsigned8	BYTE, USINT
Index 2031H Sub 1 … 252	CAN1OutSInt0 … CAN1OutSInt251	Integer8	SINT
Index 2040H Sub 1 … 126	CAN1OutWord0 … CAN1OutWord125	Unsigned16	WORD, UINT
Index 2041H Sub 1 … 126	CAN1OutInt0 … CAN1OutInt125	Integer16	INT
Index 2050H Sub 1 … 63	CAN1OutDword0 … CAN1OutDword62	Unsigned32	DWORD, UDINT
Index 2051H Sub 1 … 63	CAN1OutDInt0 … CAN1OutDInt62	Integer32	DINT

The object dictionary of interface CAN1 in total has available 16 TPDO and 16 RPDO. The first 4 TPDO and RPDO are preconfigured and activated according to the Predefined Connection Set. The first 32 Byte of input and output variables are mapped to those PDOs. Table 7 in detail lists all preconfigured PDOs for interface CAN1.

Table 7: Preconfigured PDOs for interface CAN1

PDO	CAN-ID	Data
1. RPDO	0x200 + NodeID	%MB0.0 … %MB7.0
2. RPDO	0x300 + NodeID	%MB8.0 … %MB15.0
3. RPDO	0x400 + NodeID	%MB16.0 … %MB23.0
4. RPDO	0x500 + NodeID	%MB24.0 … %MB31.0
1. TPDO	0x180 + NodeID	%MB256.0 … %MB263.0
2. TPDO	0x280 + NodeID	%MB264.0 … %MB271.0
3. TPDO	0x380 + NodeID	%MB272.0 … %MB279.0
4. TPDO	0x480 + NodeID	%MB280.0 … %MB287.0

Due to limitation to 16 TPDO and 16 RPDO, only 256 Bytes (2 * 16PDO * 8Byte/PDO) of total 504 Bytes for network variables in the marker section (2 252Bytes) can be transferred via PDO. Irrespective of that it is possible to access all variables via SDO.

The configuration (mapping, CAN Identifier etc.) of interface CAN1 typically takes place via an external Configuration Manager that parameterizes the object dictionary on the basis of a DCF file created by the CANopen configurator. By using default object inputs 1010H und 1011H, the CTR-700 supports the persistent storage and reload of a backed configuration.

Alternatively, the configuration (mapping, CAN Identifier etc.) of the static object dictionary for interface CAN1 can take place from the PLC program by using SDO function blocks. Therefore, inputs *NETNUMBER* and *DEVICE* must be used as follows:

```
NETNUMBER  := 1;            (* Interface CAN1 *)
DEVICE     := 0;            (* local Node     *)
```

The PLC program example *"ConfigCAN1"* exemplifies the configuration of interface CAN0 through a PLC program by using function blocks of type *"CAN_SDO_Xxx"*.