

# CDF600-22xx

## FIELD BUS MODULES FOR PROFINET

Supplements to the operating instructions.  
Integration into PROFINET.



This document describes the CDF600-22xx fieldbus module for PROFINET (firmware version V1.20 or higher) in both Proxy and Gateway mode. The required handling on the part of the PROFINET controller is described here using the example of a PLC of the type Siemens S7 CPU with integrated PROFINET based on Step7 V5.5. The CDF600-22xx fieldbus module for PROFINET is designated as “CDF600-22xx” in this document.

## Contents:

<b>1</b>	<b>Functions of the CDF600-22xx</b>	<b>4</b>
1.1	Range of functions in Proxy mode	5
1.2	Range of functions in Gateway mode	6
1.3	Mode overview for the CDF600-22xx	7
1.4	Overview of the communication protocols of the CDF600-22xx	8
<b>2</b>	<b>Electrical connection</b>	<b>9</b>
2.1	CDF600-2200 M12 version	9
2.2	CDF600-2201 AIDA version	9
<b>3</b>	<b>ID sensor connection</b>	<b>10</b>
3.1	Prerequisites for the safe operation of the CDF600-22xx in a system	10
3.2	ID sensor connection, e.g., bar code scanner CLV62x ... 65x	13
3.3	Connecting the CLV62x ... 65x with subnetwork (CAN bus) to the CDF600-2200 (M12 version)	14
3.4	Connecting the CLV63x ... 65x bar code scanner with heating	15
3.5	Connecting a hand-held scanner	16
3.6	Connecting other devices with RS-232 interface	17
<b>4</b>	<b>Configuration via USB on the CDF600-22xx</b>	<b>18</b>
4.1	Configuring the ID sensor via USB on the CDF600-22xx in Proxy mode	18
4.2	Parameter cloning in CDF600-22xx in Proxy mode / device replacement	21
4.3	Configuring the CDF600-22xx in Gateway mode	23
<b>5</b>	<b>Handling on the PLC side</b>	<b>26</b>
5.1	Installing the GSDML file	26
5.2	Adding sensors in HW Config	27
5.3	Assigning PN names automatically	28
<b>6</b>	<b>Data channel of the CDF600-22xx</b>	<b>29</b>
6.1	Handshake mode / Confirmed Messaging (Proxy/Gateway mode)	29
6.1.1	Using the SICK function block for PROFIBUS/PROFINET	31
6.1.2	Byte layout CDF600	31
6.1.3	Receiving data	32
6.1.4	Example 1, receipt of a single block telegram (HS mode):	33
6.1.5	Example 2, receipt of a blocked telegram (HS mode):	34
6.1.6	Send data	35
6.1.7	Example 3, transmission of a single block telegram (HS mode)	36
6.1.8	Example 4, transmission of a blocked telegram (HS mode):	38
6.1.9	Binary status bits In	39
6.1.10	Binary Status Bits Out	40
6.2	No-Handshake Mode	40
6.2.1	Byte layout in No-Handshake Mode (NH):	41
6.2.2	Receiving data in No-Handshake Mode	41
6.2.3	Example 5, receive telegram (NH mode):	42
6.2.4	Example 6, send telegram (NH mode):	43
<b>7</b>	<b>Function of the CDF600-22xx in Gateway mode</b>	<b>45</b>
<b>8</b>	<b>Digital inputs/outputs</b>	<b>47</b>
8.1	Ctrl bits	47

8.1.1	Ctrl bits In .....	48
8.1.2	Ctrl bits Out .....	49
<b>9</b>	<b>GSDML configuration in Proxy mode (optional).....</b>	<b>50</b>
<b>10</b>	<b>Appendix .....</b>	<b>51</b>
10.1	Quickstart ID sensor on the CDF600-22xx in Proxy mode via USB .....	51
10.2	Quickstart ID sensor on the CDF600-22xx in Proxy mode without USB .....	52
10.3	Troubleshooting.....	53
10.3.1	Proxy mode troubleshooting .....	53
10.3.2	Gateway mode troubleshooting .....	54
10.4	Overview of the mode switch of the CDF600-22xx .....	55
10.5	Read switch setting with GETDIAG via SOPAS terminal .....	56
10.6	Resetting the CDF600-22xx to the default settings / clearing the cloning memory.....	57
10.7	Monitoring data output via the SOPAS terminal.....	58
10.8	Function of the LEDs of the CDF600-22xx .....	59
10.8.1	Display of the Power LED.....	59
10.8.2	Status of the "SF" LED .....	60
10.8.3	Status of the "BF" LED .....	60
10.8.4	Status of the LED "EXT. IN 1".....	60
10.8.5	"P1 LNK/ACT" status LED .....	60
10.8.6	"P2 LNK/ACT" status LED .....	60
10.9	Firmware update of the CDF600-22xx (mode E) via SOPAS.....	61
10.10	Firmware update of the attached ID sensor (mode F) via SOPAS.....	62
10.11	Software versions of the CDF600-22xx.....	63
10.12	Tools for checking and assigning PN name and IP address from the PLC side .....	63
10.13	Checking and, if necessary, assigning a PROFINET name from the PLC side via HW Config..	65
10.14	Checking the PROFINET name using the PST (Primary Setup Tool).....	65
10.15	Searching the network using SOPAS Auto IP Scan .....	66
10.16	Configuration of a hand-held scanner from the IDM product family .....	67
10.17	Notes regarding operation on other PROFINET controllers.....	68
10.18	Changes Firmware version V1.20 .....	68

# 1 Functions of the CDF600-22xx







The CDF600-22xx complies with PROFINET Conformance Class B.

The following features are supported:

- Cyclic RT communication.
- Automatic address issue for equipment replacement without engineering tool.
- I&M 0 functionality as well as reading and writing from I&M 1-4
- Ring support – Media Redundancy Protocol (MRP Client)
- FAST Ethernet 100 Base TX/FX
- SNMP support
- Neighborhood detection LLDP

## 1.1 Range of functions in Proxy mode

The Proxy mode (rotary coding switch “Mode” to 0) can be used for the ID sensors listed below. In Proxy mode, the GSDML file of the respective ID sensor must be used.

ID sensor	Model name / Version	GSDML file to be used (or newer version)
 SICK CLV61x	CLV61x-Fxxx / V1.21 or higher	GSDML-V2.3-SICK-CLV61x_via_CDF600-20131029.xml
 SICK CLV62x-65x	CLV62x-65x / V5.26 or higher	GSDML-V2.3-SICK-CLV62xCLV65x_via_CDF600-20130730.xml
 SICK CLV69x	CLV69x / V1.60 or higher	GSDML-V2.3-SICK-CLV69x_via_CDF-20150312.xml
 SICK LECTOR 62x	Lector62x / V2.10 or higher	GSDML-V2.3-SICK-Lector62x_via_CDF-20150312.xml
 SICK RFH6xx	RFH6xx / V3.11 or higher	GSDML-V2.3-SICK-RFH6xx_via_CDF600-20150312.xml
 SICK RFU6xx	RFU6xx / V1.62 or higher	GSDML-V2.3-SICK-RFU6xx_via_CDF600-20140605.xml

In this mode, the complete range of functions of the CDF600-22xx is available:

- Communication with the PROFINET/PLC (data channel for sending and receiving). Depending on the ID sensor, communication can occur in CDF mode and CDF-NoHandshake mode.
- The “EXT. IN1” switching input can be used directly to trigger the ID sensor.
- Ctrl bits for the PROFINET/PLC (according to the digital switching inputs/outputs on the ID sensor and EXT. IN1)
- The USB connection can be used to configure and monitor the ID sensor via SOPAS. (The connection is implemented on the serial AUX port of the ID sensor.)
- Cloning parameter integrated in CDF600-22xx (same as parameter CMC600 memory module)
- Configuration of the ID sensor from the PLC via GSDML file (depending on the ID sensor)
- Configuration of the ID sensor via PROFINET with SOPAS when the PC is directly connected to the network (Ethernet TCP/IP connection) via the IP address of the CDF600-22xx via port 2111/2112. In the CDF600-22xx, this is implemented on the serial AUX interface of the ID sensor. The ID sensor in SOPAS is also displayed during the AutoIP scan (for IP addresses that are not 0). It is recommended for Lector6xx to connect SOPAS directly with the Lector via USB or Ethernet, since the configured connection via the CDF600-2 cannot display any images.

The following triggering types of the ID sensor are possible:

- Triggering by photoelectric sensor or hardware signal at input “EXT. IN1” (configure ID-sensor to EXT. IN1) or
- Triggering by software trigger (configure ID sensor to command)
- Triggering by trigger bit in the Ctrl bits (configure ID sensor to fieldbus input)

If the ID sensor supports it, it can also be operated in free-running mode.


## 1.2 Range of functions in Gateway mode

In Gateway mode (rotary coding switch “Mode” to 2 or 4), any device can be connected that can generate the required communication parameters such as RS-232, 57.6 (position 2) or 9.6 kBd (position 4), the data format 8 data bits, no parity, 1 stop bit) and an STX/ETX framing, **e.g., hand-held scanner of the IDM or Lector65x product family.**

In Gateway mode, the range of functions of the CDF600-22xx is limited:

- Communication with the PROFINET/PLC (data channel for sending and receiving).  
Only CDF mode **with handshake** is available.  
From V1.20 and higher there is also the option to set the **NoHandshake** mode via SOPAS in CDF600-2. The SDD file required for this can be downloaded via SOPAS from mysick.com. A SDD upload from CDF600-2 is not possible.
- The PLC can detect the input “EXT. IN1” via the Ctrl bits.
- Via **USB** and via **Ethernet Port 2111** only the CDF600-2 itself can be configured with SOPAS.. Port 2111 is connected logically with the CDF600-2.
- The USB connection **cannot** be used to configure the ID sensor.
- A SOPAS-capable ID sensor can be configured via **Ethernet Port 2112**. Port 2112 is implemented on the serial AUX port of the ID sensor.
- There is no parameter cloning function for the connected ID sensor

In this mode, the GSDML file of the CDF600-22xx must be used:

Sensor	Model name / Version	GSDML file to be used (or newer version)
	CDF600-2 / V1.20 or higher	GSDML-V2.3-SICK-CDF600-20150312.xml

The CDF600-22xx is a PROFINET gateway for transfer from RS-232 to PROFINET. It has a serial RS-232 interface that can connect to an ID sensor as follows:

- 57.6 or 9.6 kBd
- 8, n, 1
- Data must be framed by STX (Hex 02) and ETX (Hex 03).  
From V1.20 and higher there is also the option to switch the serial protocol to Cola B (binary) via SOPAS. The SDD file required for this can be downloaded via SOPAS from mysick.com. A SDD upload from CDF600-2 is not possible. Also to be noted is that the connected sensor must also use this protocol, e.g., this can be an MSC800 as of V3.40 and higher.  
In this documentation it is assumed that ColaA, i.e., the standard STX/ETX frame will be used, unless it is specially marked.
- Data can be sent and received

The interface is connected to the AUX interface of the ID sensor (pins 2, 3 and 5 of the 15-pin HD socket).

If triggering should be carried out from the PLC, a software trigger must be used.

The PLC can detect the input “EXT. IN1” via the Ctrl bits, but the input cannot be used for direct triggering of the ID sensor.

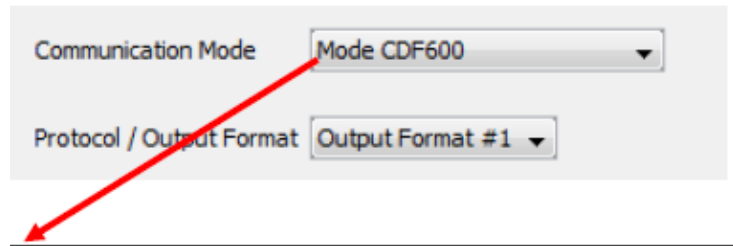
If the ID sensor supports it, it can also be operated in free-running mode.

### Important!

A separate connection box can be used to integrate a bar code scanner of the CLV42x ... 49x product family or 2D code reader of the ICR84x-2/85x product family and the host interface of the device can then be connected to the CDF600-22xx. The CDF600-22xx must then be operated in Gateway mode. For electrical connection, see chapter 2.

### 1.3 Mode overview for the CDF600-22xx

Operation of the CDF600-22xx depends on the position of the “Mode” rotary coding switch and the communication protocol set in the ID sensor. The switch is located under the cover.



CDF600-22xx “Mode” switch	Operating mode	Communication modes (configured in the sensor)	Data channel		Digital I/Os
			Handshake / Confirmed messaging (CM) Operation of the SICK function blocks possible	No-Handshake	
Mode 0 →	Proxy	CDF600 (57,600 baud)	X		Ctrl bits
		CDF600 NoHandshake (57,600 baud)		X	Ctrl bits
Mode 2 →	Gateway	CDF600 (57,600 baud)	X		Ctrl bits
Mode 4 →		CDF600 (9,600 baud)	X		Ctrl bits

Table: CDF600-22xx operating modes

In Gateway mode, the communication protocol is fixed to CMF600 (with handshake) and only be changed via SOPAS on No-Handshake. In Gateway mode, no GSDML configuration is permitted.

Operation of the SICK function blocks for PROFIBUS / PROFINET is only possible in Handshake mode / Confirmed Messaging (CDF600).

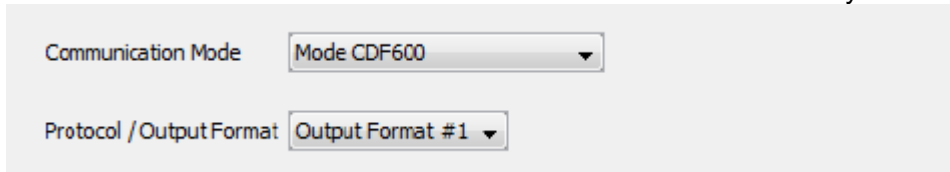
In Proxy mode, the communication protocol can be configured in the ID sensor. Depending on the sensor, various communication protocols are supported:

Proxy-capable ID sensors	Firmware version	Communication protocol		GSD configuration
		CDF600	CDF600 No-Handshake	
CLV61x-FIELDBUS (not CLV61x-Cxxx or CLV61x-Sxxx)	V1.21 or higher	X	X	X
CLV62x ... 65x	V5.26 or higher	X	X	X
CLV69x	V1.60 or higher	X	X	---
Lector62x	V2.10 or higher	X	X	X
RFH6xx	V3.11 or higher	X	X	X
RFU6xx	V1.62 or higher	X	X	X

## 1.4 Overview of the communication protocols of the CDF600-22xx

### In Proxy mode (mode 0):

The CDF600-22xx offers two communication protocols. The communication protocol can be set in the ID sensor under “Parameter / Network Interfaces IOs / Fieldbus Gateway”:



The screenshot shows a configuration window with two dropdown menus. The first dropdown, labeled 'Communication Mode', is set to 'Mode CDF600'. The second dropdown, labeled 'Protocol / Output Format', is set to 'Output Format #1'.

### **CDF600 Handshake mode / Confirmed Messaging** (default, recommended)

- Data channel compatible with the Byte Handshake mode of the CDF600 PROFIBUS, CMF400 PROFIBUS, CDM425, and PROFINET on Board.
- Send and receive with max. telegram length of 4000 bytes (with blocking)
- A handshake is required on the PLC side.
- The SICK function blocks can be used for PROFIBUS / PROFINET
- Ctrl bits can be used to trigger or to set I/O's and to monitor.

### **CDF600 No Handshake Mode:**

- Data channel compatible with the No-Handshake mode of the CDF600 PROFIBUS, CMF400 PROFIBUS, CDM425, and PROFINET on Board
- The max. telegram length is limited by the size of the input/output range and is max. 123 bytes. There is no fragmenting / blocking.
- No **handshake** is required on the PLC side.
- Ctrl bits can be used to trigger or to set I/O's and to monitor.

### In Gateway mode (mode 2 or 4):

In Gateway mode, the communication protocol is fixed to CMF600 (with handshake) and only be changed via SOPAS on No-Handshake.

### **CDF600 Handshake mode / Confirmed Messaging**

- Data channel compatible with the Byte Handshake mode of the CDF600 PROFIBUS, CMF400 PROFIBUS, CDM425, and PROFINET on Board
- Send and receive with max. telegram length of 4000 bytes (with blocking)
- A handshake is required on the PLC side.
- Ctrl bits may be used. However only the DevReady bit and ExtIn1 are allocated.

### **CDF600 No Handshake Mode:**

- Data channel compatible with the No-Handshake mode of the CDF600 PROFIBUS, CMF400 PROFIBUS, CDM425, and PROFINET on Board
- The max. telegram length is limited by the size of the input/output range and is max. 123 bytes. There is no fragmenting / blocking.
- No **handshake** is required on the PLC side.
- Ctrl bits may be used. However only the DevReady bit and ExtIn1 are allocated.



## 2 Electrical connection

The electrical connection incl. pin assignment and wire colors of the cables can be found in the operating instructions for CDF600-2200 (8015921) and the CDF600-2201 (8016852) or online at [www.sick.com/CDF600-2](http://www.sick.com/CDF600-2).

### 2.1 CDF600-2200 M12 versione



### 2.2 CDF600-2201 AIDA version



### 3 ID sensor connection

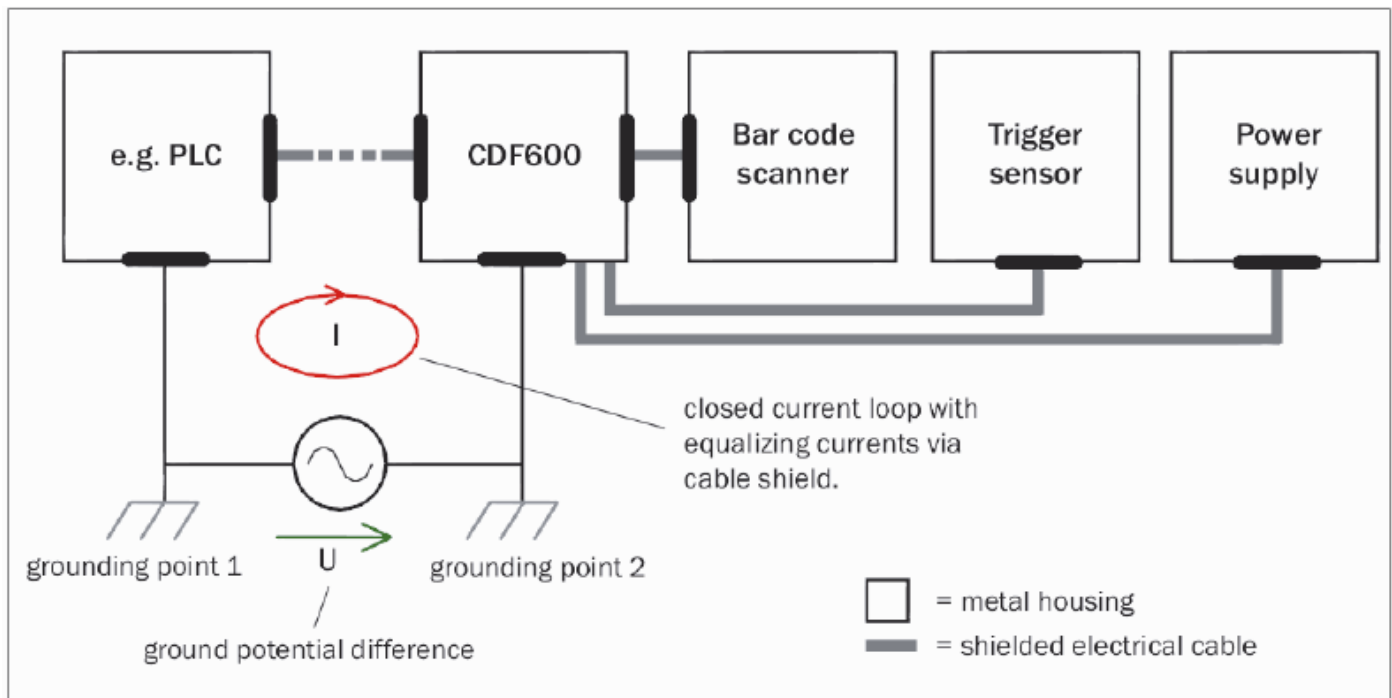
#### 3.1 Prerequisites for the safe operation of the CDF600-22xx in a system

The CDF600-22xx fieldbus module is connected to other peripheral devices, such as bar code scanners, read cycle sensor(s), PLC/host, power supplies, via screened cables. The cable shields lie on the metal housing of the CDF600-22xx. The device can either be connected to the system ground via the elongated holes or the shield of the power supply cable, for example.

If the peripheral devices have metal housings and if the cable shields also lie on their housings, it is assumed that all devices involved in the installation have the same ground potential. This is ensured by:

- Mounting the devices on conductive metal surfaces
- Correctly grounding the devices/metal surfaces in the system
- If necessary, low-impedance and current carrying equipotential bonding between areas with different ground potentials.

If these conditions are not met, e.g., on devices in a widely distributed system over several buildings, potential equalization currents may, due to different ground potentials, flow along the cable shields between the devices.



Due to insufficient ground potential equalization, voltage differences arise between the grounding points 1 and 2. The current loop closes via the screened cables and housing.

#### Risk of injury/risk of damage via electrical current

Potential equalization currents between the CDF600-22xx and other grounded devices in the system can have the following effects:

- Dangerous voltages on the metal housing, e.g., of the CDF600-22xx
- Incorrect function or irreparable damage to the devices
- Damage/irreparable damage of the cable shield due to heating and cable fires

Where local conditions are unfavorable and thus do not meet conditions for a safe earthing method (same ground potential at all grounding points), take measures in accordance with the following formats.

#### Remedial measures

The most common solution to prevent potential equalization currents on cable shields is to ensure low-impedance and current carrying equipotential bonding. If this is not possible, the following solution approaches serve as a suggestion.

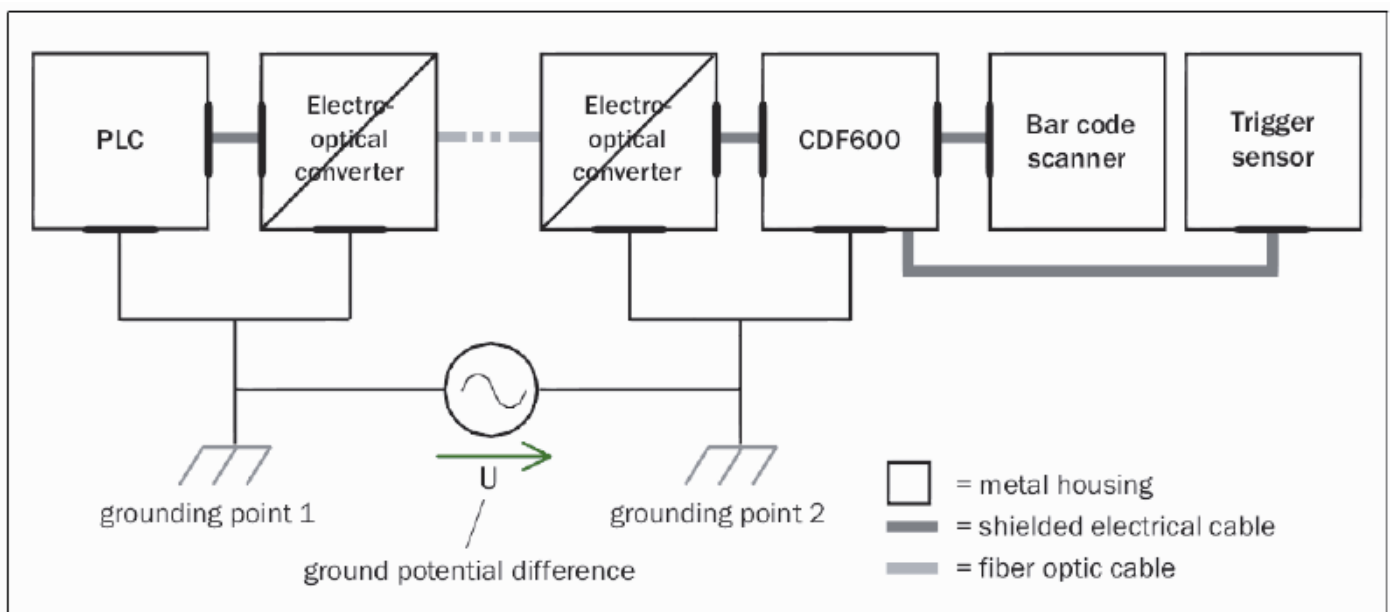
### Important!

We expressly advise against opening up the cable shields. This would mean that the EMC limit values can no longer be complied with and that the safe operation of the device data interfaces can no longer be guaranteed.

#### a) Measures for widely distributed system installations

On widely distributed system installations with correspondingly large potential differences, we recommend setting up local islands and connecting them using commercially available electro-optical signal isolators. This measure achieves a high degree of resistance to electromagnetic interference while at the same time complying with all the requirements of EN 60950-1.

The following figure shows how these measures work:

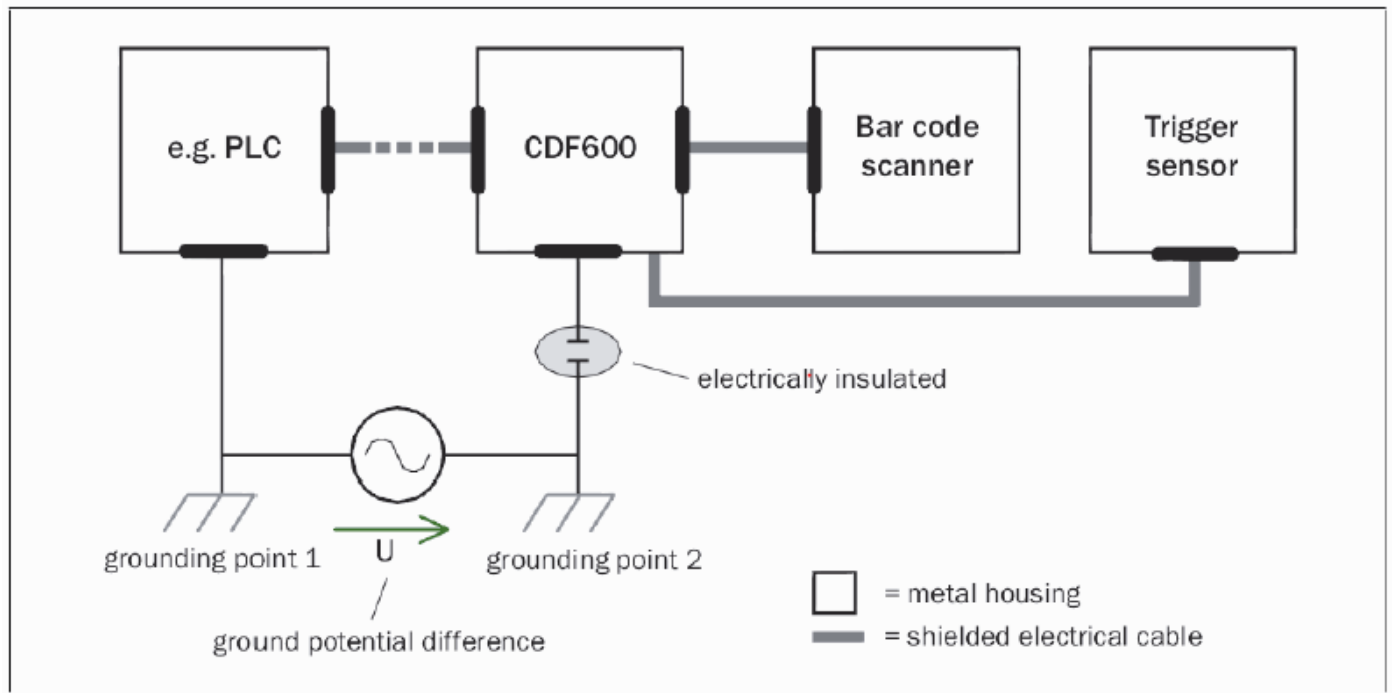


The ground loop is isolated by using the electro-optical signal isolator between the islands. Within the islands, a stable equipotential bonding prevents equalizing currents on the cable shields.

#### b) Measures for small system installations

For smaller installations with only slight potential differences, insulated installation of the CDF600-22xx and of peripheral devices may be a sufficient solution.

The following figure shows how these measures work:



Even in the event of large differences in the ground potential, ground loops are effectively prevented. As a result, equalizing currents can no longer flow via the cable shields and metal housing.

### Important!

The power supply of the CDF600-22xx and the connected peripheral devices must also guarantee the required level of insulation.

Under certain circumstances, a tangible potential can develop between the insulated metal housings and the local ground potential.

### 3.2 ID sensor connection, e.g., bar code scanner CLV62x ... 65x


CLV6xx and other ID sensors with fixed connecting cable and 15-pin D-Sub HD plug can be connected to the CDF600-22xx directly. The max. length of the cable is 5 m incl. possible extension cables.



ID sensors with 12-pin M12 plug connector (e.g., CLV62x ... 65x – Ethernet version) can be connected using one of the following cables:

	Cable, M12, 12-pin, to connection module CDx 15-pin D-sub, 5 m (socket/plug)	Connecting cable (plug-socket)	2042915
	Cable, M12, 12-pin, to connection module CDx 15-pin D-sub, 0.9 m (socket/plug)	Connecting cable (plug-socket)	2042916
	Cable, M12, 12-pin, to connection module CDx 15-pin D-sub, 3 m (socket/plug)	Connecting cable (plug-socket)	2042914
	Cable, M12, 12-pin, to connection module CDx 15-pin D-sub, 2 m (socket/plug)	Connecting cable (plug-socket)	2041834
	Cable, M12, 12-pin, to connection module CDx 15-pin D-sub, 3 m (socket/plug), drag-chain compliant	Connecting cable (plug-socket)	2061604

ID sensors with 17-pin M12 plug connector (e.g., Lector62x or RFU6xx – Ethernet version) can be connected using one of the following cables:

	Cable, M12, 17-pin, to connection module CDx 15-pin D-sub, 2 m (socket/plug)	Connection cable (plug-socket)	2055419
	Cable, M12, 17-pin, to connection module CDx 15-pin D-sub, 3 m (socket/plug)	Connection cable (plug-socket)	2055420
	Cable, M12, 17-pin, to connection module CDx 15-pin D-sub, 0.9 m (socket/plug)	Connection cable (plug-socket)	2049764
	Cable, M12, 17-pin, to connection module CDx 15-pin D-sub, 0.35 m (socket/plug)	Connecting cable (plug-socket)	2056184
	Cable, M12, 17-pin, to CDx 15-pin D-sub, 5 m (socket/plug)	Connecting cable (plug-socket)	2055859
	Cable, M12, 17-pin, to connection module CDx 15-pin D-sub, 3 m (socket/plug), drag-chain compliant	Connecting cable (plug-socket)	2061605

### 3.3 Connecting the CLV62x ...65x with subnetwork (CAN bus) to the CDF600-2200 (M12 version)

The CAN bus can be used to connect additional ID sensors to the CLV6xx. The devices operate together as a master/slave or mux/server configuration.

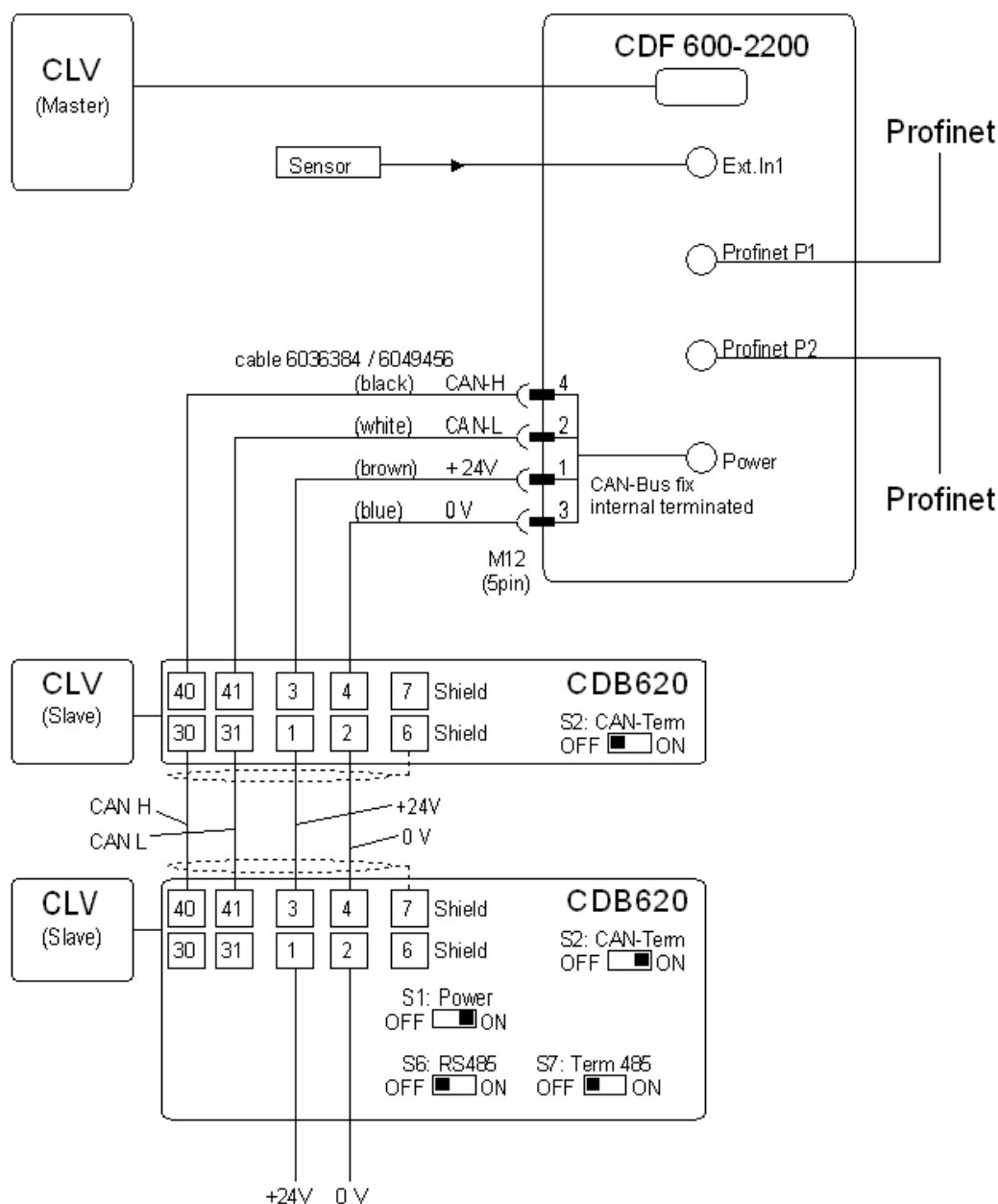
CAN bus:

Master / slave: max. 5 devices

Mux / server: max. 5 devices

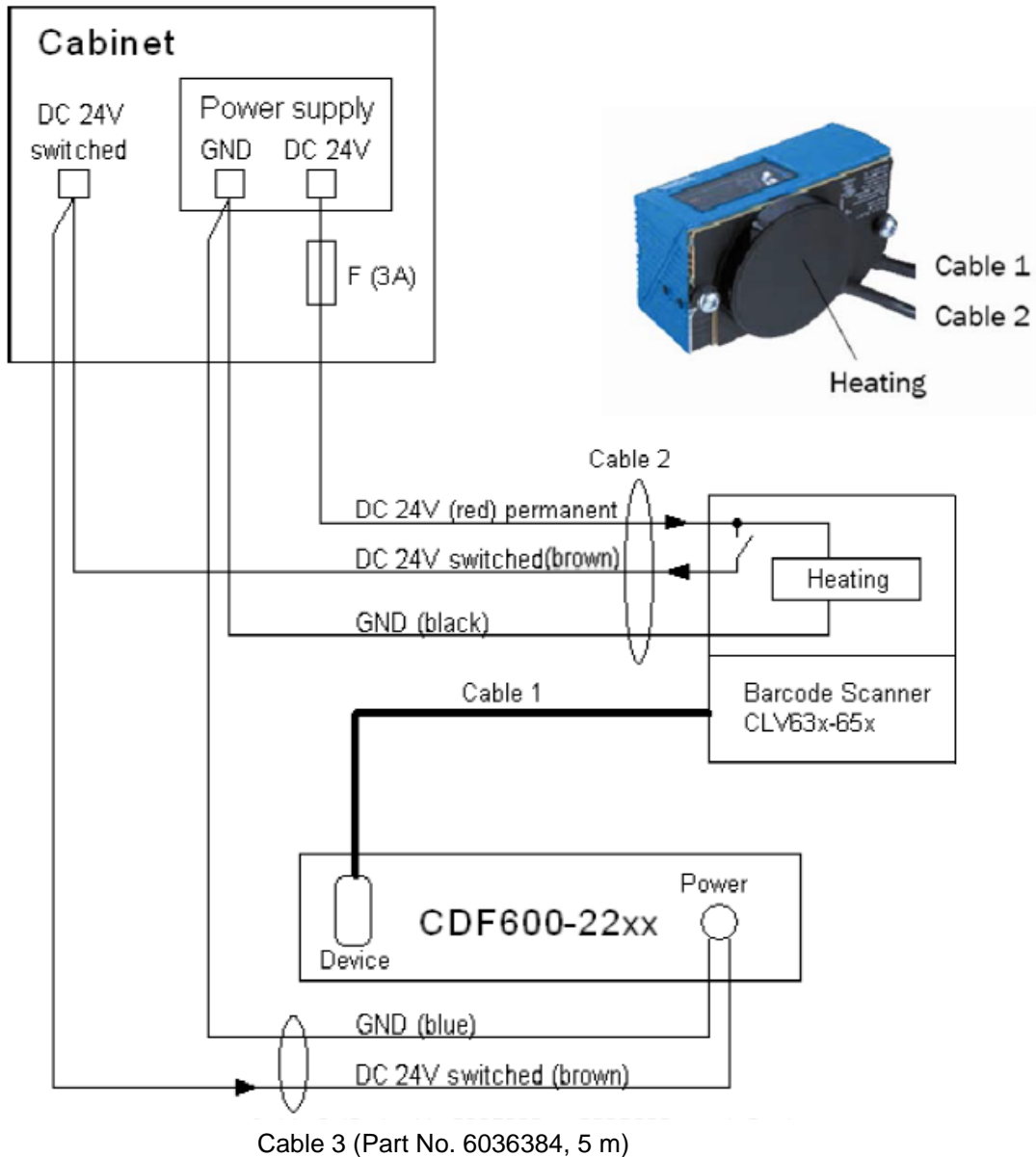
The CDF600-2200 (M12 version) offers the option of connecting the CAN bus via the power plug.

Connect: **CLV 62x-65x** as **CAN-Scanner-Network** to **CDF600-2200 PROFINET**



### 3.4 Connecting the CLV63x ... 65x bar code scanner with heating

The supply voltage feed (24 V DC  $\pm$  10%) must be protected on the user-side by a 3 A fuse in the control cabinet. A separate fuse must be used for each unit consisting of a fieldbus module and bar code scanner with heater.



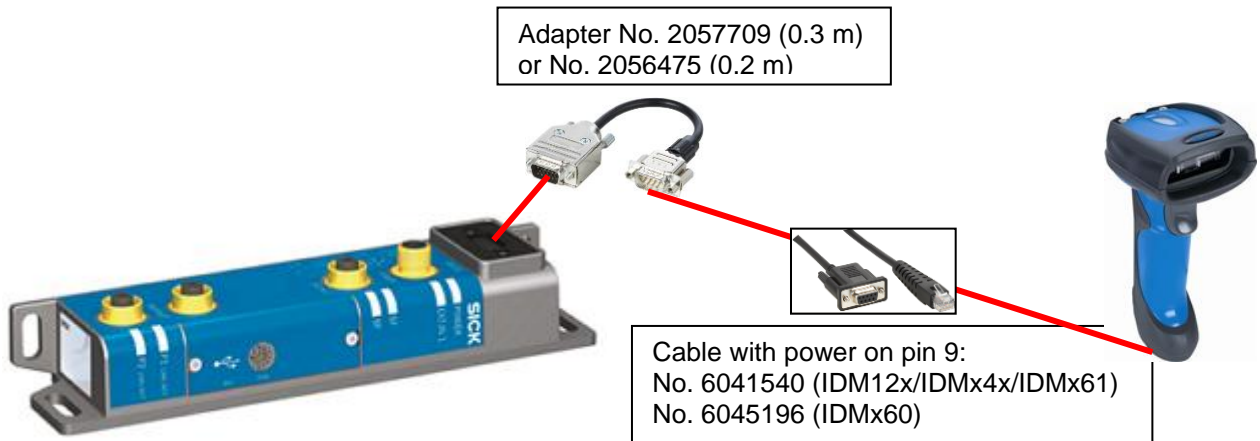
The supply voltage for the fieldbus module is routed via a temperature-dependent switch for the bar code scanner's heating. After a warm-up phase of approx. 40 min (at  $-35^{\circ}\text{C}$  and 24 V DC), the temperature-dependent switch releases the supply voltage for the fieldbus module. The fieldbus module now supplies the electronics of the bar code scanner.

#### Cable lengths

- The supply cable (cable 3) from the 24 V DC switched/power supply unit (GND) to the fieldbus module must not exceed 5 m.
- The supply cable (cable 2) from the power supply to the heating for the bar code scanner is 2 m and must not be extended.

### 3.5 Connecting a hand-held scanner

Connection of a hand-held scanner of the IDM product family in Gateway mode using an adapter via RS-232. The adapter includes a voltage converter from DC 24 to 5 V for the voltage supply of the hand-held scanner, which means that a separate plug-in power supply is not necessary.



For information on how to configure the hand-held scanner: See appendix, chapter 10.16

Set the CDF600-22xx to mode 2.

The CDF600-22xx is then fixed to 57600 baud and the communication protocol is set to CDF600 (see chapter 1.3)

Data can be received via SICK function block (data receipt only).

**Note:**

The S7 function block for CLV6xx/LECTOR®62x PROFINET/PROFIBUS can be used for data receipt. Optional actions such as trigger or COM test cannot be used. Likewise, if error message No. 9 or No. 1 occurs, this can be ignored. This does not affect the data receipt from the hand-held scanner.



### 3.6 Connecting other devices with RS-232 interface

If the CDF600-22xx is operated in Gateway mode, other devices with RS-232 interface can be connected. The following communication parameters are required:

- RS-232 data interface with 57.6 or 9.6 kBd transmission rate
- Data format 8 data bits, no parity, 1 stop bit
- Data must be framed by STX (Hex 02) and ETX (Hex 03)

**Receive data** (e.g. read results from sensor to PLC):

The data string must always be framed with STX/ETX.

**Send data**(e.g., commands from PLC to sensor):

The CDF600-22xx always adds the STX/ETX frame.

This cannot be suppressed.

CDF600-22xx

15-pin HD socket device:

External RS-232 device:

TxD, pin 2 ----->-----	RxD
RxD, pin 3 ----<-----	TxD
Gnd, pin 5 -----	Gnd

If an ID sensor (e.g., CLV6xx, Lector62x, RFH6xx, RFU6xx, etc.) is connected to the CDF600-22xx and operated in Gateway mode, the serial **AUX interface** of the ID sensor is connected to the CDF600-22xx. The serial AUX interface may need the appropriate output format with STX/ETX frame.

In Gateway mode, the USB interface of the CDF600-22xx cannot be used to configure the connected ID sensor.

The ID sensor must then be configured, for example, via a direct Ethernet interface or USB.

In Gateway mode the PLC can detect the “EXT. IN 1” input via the Ctrl bits but the input cannot use it for direct triggering of the ID sensor.

If necessary the “EXT. IN1” input detected via the Ctrl bits. can be used to send a software trigger to the sensor from the PLC.

## 4 Configuration via USB on the CDF600-22xx

### 4.1 Configuring the ID sensor via USB on the CDF600-22xx in Proxy mode

If the CDF600-22xx is operated in Proxy mode (rotary coding switch at position 0), a USB cable can be used to configure the ID sensor via SOPAS V2.36 (or higher). The Micro-B USB socket is located under the cover for the rotary switch.

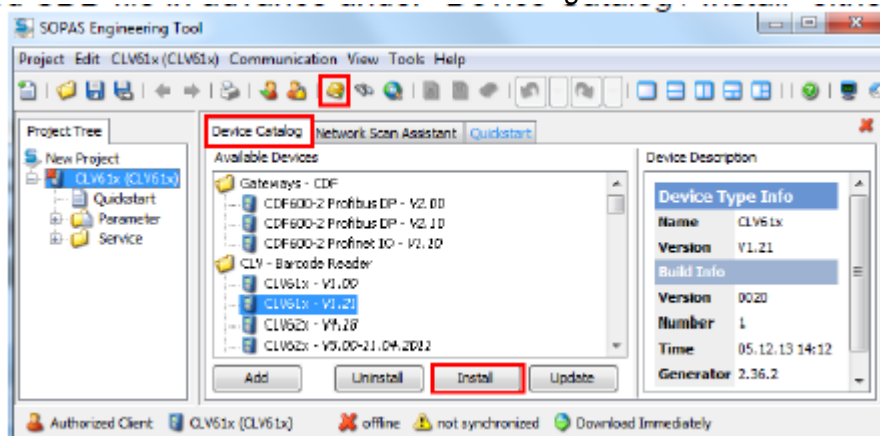


Select the network wizard in SOPAS and connect USB. SOPAS shows the USB connection of the CDF600-22xx as the communication interface, **but the attached ID sensor itself is recognized as the device** and not the CDF600-22xx.

Detected Devices	Communication Interface	Suitable Device Descriptions
CLV62x (station51)	USB - CLV62x - 12020272	★ CLV62x - V5.26

The USB interface is implemented by the CDF600-22xx on the serial AUX interface (RS-232) of the ID sensor. No images, for example, can be transferred from the LECTOR®62x. If available, of course, a **direct Ethernet or USB interface** of the ID sensor can be used for configuration or to transfer images.

In the SOPAS Network Scan Assistant, **USB activated** is required to search. The required **SDD file of the ID sensor** must be installed on the PC under SOPAS. It can take a relatively long time depending on the ID sensor if not installed and the SDD file has to be loaded from the device. It is recommended to install the required SDD file in advance under “Device Catalog / Install” either online or by data transfer.



If the USB connection is established, SOPAS or SOPAS Single Device launches automatically. To use separately, SOPAS ET must be launched separately and the connection to the ID sensor re-established using the network scan assistant via USB.

For the CDF600-22xx, no direct configuration is required, **i.e., the connected ID sensor automatically recognizes the CDF600-22xx**. SOPAS shows this as a system status in the “System Information” display window of the ID sensor as “CDF600 detected”. The firmware version of the CDF600-22xx is also displayed.

System Information							
Type	First time	Last time	Description	Info	State	Counter	Number
Information	8:196:39:00	8:196:39:00	Reset by Power On			1	0x200001E
Information	0:03:00	0:03:00	CDF600 detected	Firmware: V1.10.66		1	0x200070A

In the configuration, the option “Profinet Proxy CDF600” automatically appears under the device page “Network / Interfaces / IO / Fieldbus Gateways”:

The **PROFINET station name** (=PN Station Name) is displayed here and can be changed. The PROFINET station name can also be changed by the PLC. See appendix, chapter 10.12.

This PROFINET station name is also used as the device name in the default settings. The device name is displayed in the parameter tree in SOPAS and helps when searching for devices. If required, this function can be deactivated using the “Use 'PN Station Name' also for 'Device Name'” check box.

If the CDF600-22xx is **not yet active** in the PROFINET - data exchange to the IO Controller (PLC), the PN IP-Address, PN Subnet-Mask and the PN Default Gateway of the CDF600-22xx can also be set here. However, note that this is normally carried out automatically by the IO controller (PLC) if the PN station name matches. If the IP is automatically assigned by the IO controller (PLC), the IP address is temporarily accepted and then displayed as not remanent in SOPAS.

Remanent: (IP address)

This indicates whether the PN IP address, subnet and gateway are permanently stored and also immediately available once power is restored or whether these have only been temporarily assigned and the IP address is 0.0.0.0 following a restart.

Remanent: (PN station name):


This indicates whether the station name has been permanently stored.

PN state: connected → The CDF600-22xx with ID sensor is exchanging data with PROFINET. This corresponds to the “BF” LED = off.

After changing the PN name or IP address, click “Apply” to use the new settings.


The **Communication mode for the CDF600-22xx** can also be set here. If changed from the default setting, the parameters must be **saved permanently** and the CDF600-22xx restarted. For details about the modes of communication, see chapter 1.3.

Likewise, the IM data, such as Plant designation, Location designation, Installation date (mandatory format yyyy-mm-dd hh:mm) and Additional information, can also be displayed and set here. This data is only used for organization and can be read and written by the IO controller (PLC) or by the higher-level system.



Plant designation

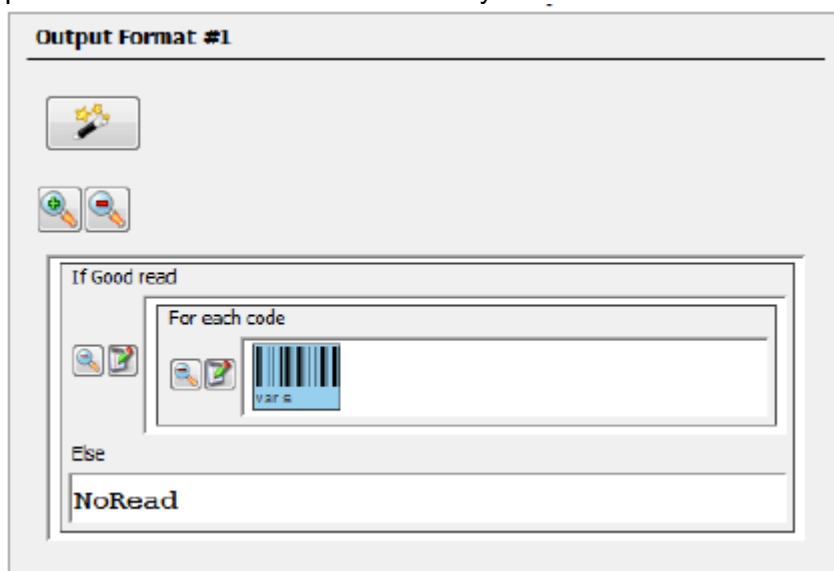
Location designation

Installation date 




Additional information

The required output format must also be set. The format does not require any STX/ETX frame because transport works in Proxy mode without this ID. If STX/ETX is set as the frame, the characters are entered in the data for the PLC as control characters, which is not normally desirable.

Set output format **without** STX/ETX in Proxy mode:




**Output Format #1**

If Good read

For each code

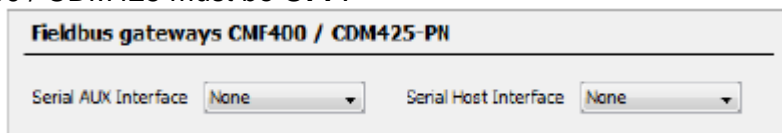


Else

NoRead

**Warning:**

The “CDF600 Proxy” SOPAS configuration block is solely responsible for operation of the CDF600-22x. Additional configuration of other fieldbus gateways must be avoided. The connectable fieldbus gateway CMF400 / CDM425 must be **OFF**:



**Fieldbus gateways CMF400 / CDM425-PN**

Serial AUX Interface  Serial Host Interface

Only **one fieldbus sensor** must be operated simultaneously in the ID sensor.

## 4.2 Parameter cloning in CDF600-22xx in Proxy mode / device replacement

The CDF600-22xx also features a cloning parameter memory for the ID sensor in Proxy mode. The function is identical to the CMC600.

When **switched on**, the ID sensor checks whether a valid, matching parameter set is stored on the CDF600-22xx. If it is, then this is also stored and used on the ID sensor. If the parameter set on the CDF600-22xx is empty when it is switched on (delivery state), then the ID sensor stores its current parameters on the CDF600-22xx. When it is next switched on, the ID sensor loads this again from the CDF600-22xx.

During the configuration of the ID sensor, the parameters, incl. PN name, are stored on the ID sensor and **also on the CDF600-22x every time they are stored permanently**. As a result the storage process takes longer. In this way the parameter set on the CDF600-22xx is synchronized with the ID sensor.

### ID sensor replacement:

If the ID sensor is replaced with a sensor of the same type, it is ready for operation as soon as it is switched on, as all parameters, incl. PN name, are transferred directly from the CDF600-22xx to the ID sensor. The previous configuration of the exchange unit (ID sensor) is not relevant and is overwritten.

### CDF600-22xx replacement:

If the CDF600-22xx is replaced with a device of the same type, it is important that the new CDF600-22xx: does not contain any cloning parameters. If necessary, this can be checked by using the GETDIAG via Terminal command in mode 2.

See appendix, chapter 10.5. See also “Clear cloning memory” on the next page or in the appendix, chapter 10.6

If the parameter set on the CDF600-22xx is empty when it is switched on (delivery state), then the ID sensor stores its current parameters on the CDF600-22xx. When it is next switched on, the ID sensor loads this again from the CDF600-22xx.

### CDF600-22xx and ID sensor replacement:

If both are replaced and both components have default settings, the PN name is empty.

If automatic name assignment has been activated in PN and the topology has been taught-in, the station is assigned the PN names by the PROFINET controller (PLC).

Normally, the IP address is then also assigned. The CDF600-22xx is then active in PN.

Configuration can then be carried out either via SOPAS through USB or via TCP/IP (IP address of the CDF600-22xx, port 2111 or 2112) or a GSDML configuration is stored.

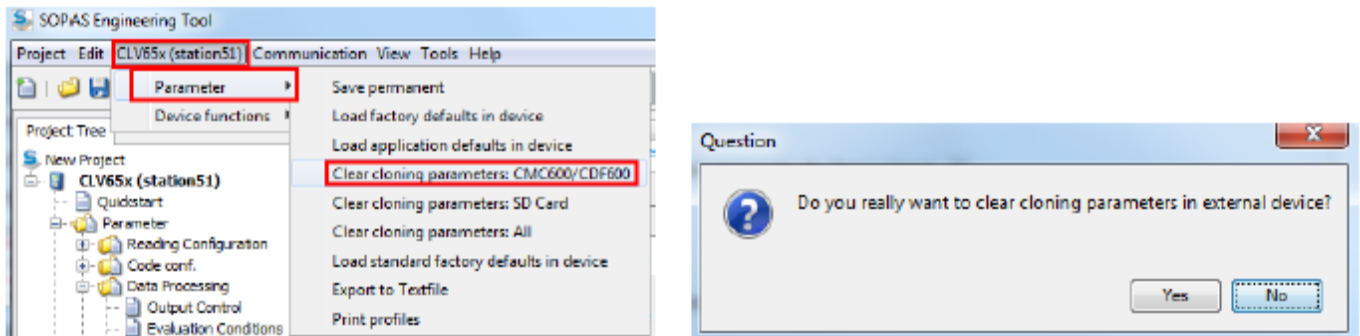
### GSDML configuration:

If a GSDML configuration is stored in the HW Config of the PROFINET Controller (PLC), then configuration is always carried out automatically whenever the PROFINET is started. This overwrites the existing parameters.

An existing **GSDML configuration therefore always overrides any parameters**. This cannot be suppressed locally in the ID sensor or the CDF600-22xx.

**Clearing the cloning memory in the CDF600-22xx via SOPAS:**

In Proxy mode, the cloning memory in the CDF600-22xx can be cleared in SOPAS when configuring the ID sensor via “CLV ... / Parameter / Clear cloning parameters: CMC600/CDF600”. A prompt then appears.



Alternatively, the cloning memory in CDF600-22xx can be cleared in mode 2 via a command.

See appendix, chapter 10.6.

The cloning memory can also be checked using the GETDIAG via Terminal command in mode 2. See appendix, chapter 10.5.

### 4.3 Configuring the CDF600-22xx in Gateway mode

In Gateway mode (rotary coding switch set to 2 or 4), it is **not necessary to configure the CDF600-22xx via SOPAS**. Both the PROFINET station name and the IP address can be assigned by the PLC. See appendix, chapter 10.12. This is also normally recommended.

**Alternatively**, the PROFINET station name and the IP address can also be assigned and checked by SOPAS.

If the CDF600-22xx is operated in Gateway mode and the PC is connected via USB, **the CDF600-22xx itself appears as a device** in SOPAS during scanning.

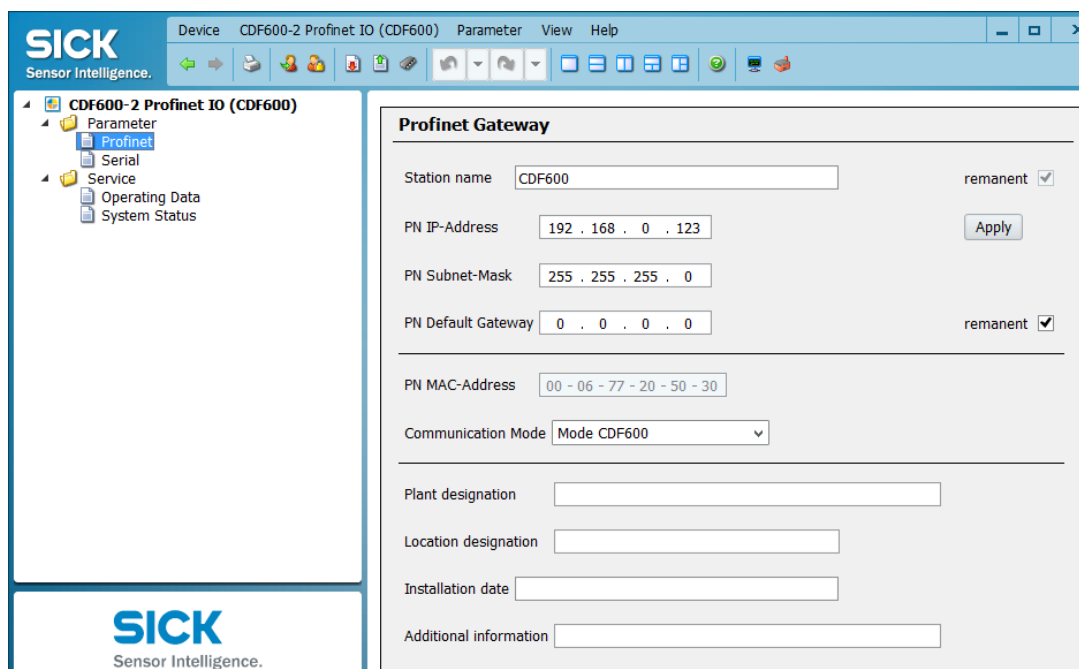
To display the device parameters, the **SDD file of the CDF600-22xx must be loaded in SOPAS in advance**. If missing, the SDD file cannot be loaded from the CDF600-22xx.

The **PROFINET station name** can be displayed and changed on the “Parameter” tab. This can also be changed from the PLC.

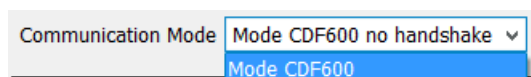
If the CDF600-22xx is **not yet active** in the PROFINET - data exchange to the IO Controller (PLC), the PN IP-Address, PN Subnet-Mask and the PN Default Gateway of the CDF600-22xx can also be set on the “Parameter” tab.

However, note that this is normally carried out automatically by the IO controller (PLC) if the PN station name matches. If the IP is automatically assigned by the IO controller (PLC), the IP address is temporarily accepted and then displayed as not remanent in SOPAS.

SOPAS CDF600-22xx / Parameter:

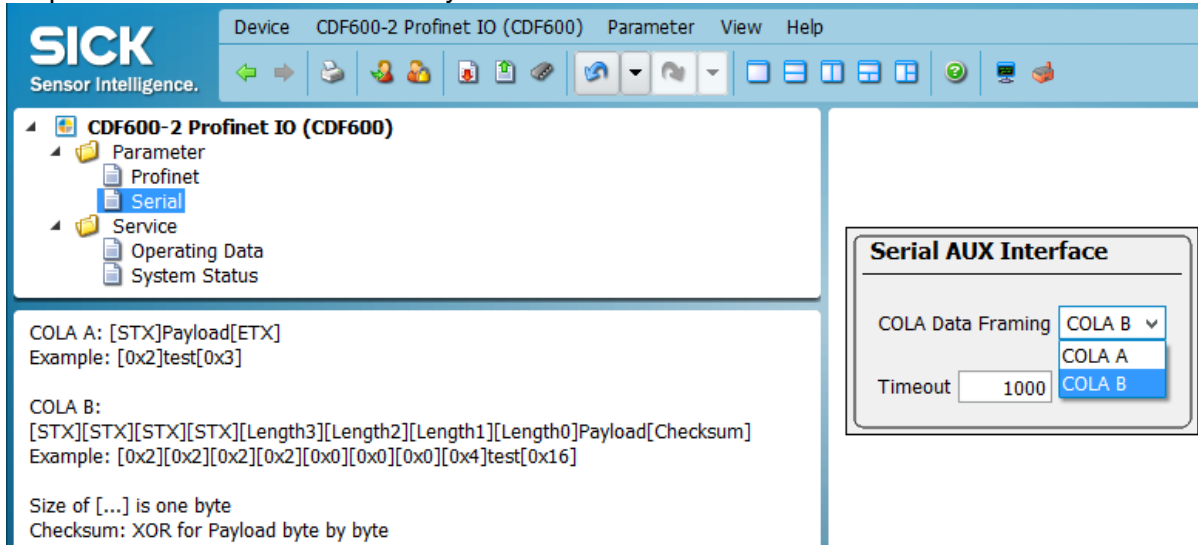


Here it can be switched to NoHandshake mode under communication protocol via SOPAS. The default is “CDF600” with handshake.



Likewise, the IM data, such as Plant designation, Location designation, Installation date (mandatory format yyyy-mm-dd hh:mm) and Additional information, can also be displayed and set here. This data is only used for organization and can be read and written by the IO controller (PLC) or by the higher-level system.

The serial protocol can be set in Gateway mode via SOPAS under “Parameter / serial”:



**ColaA** is the default setting and complies with the standard STX ETX frame.

No STX/ETX may not be in the user data itself.

In further documentation it is assumed that ColaA, will be used, unless it is specially marked.

**ColaB** is a binary protocol that works with length specifications and checksum. The MSC800 from V3.40 and higher can set this protocol on the serial host interface and enables with this a transparent transfer of all data values from 00 to FF hex.

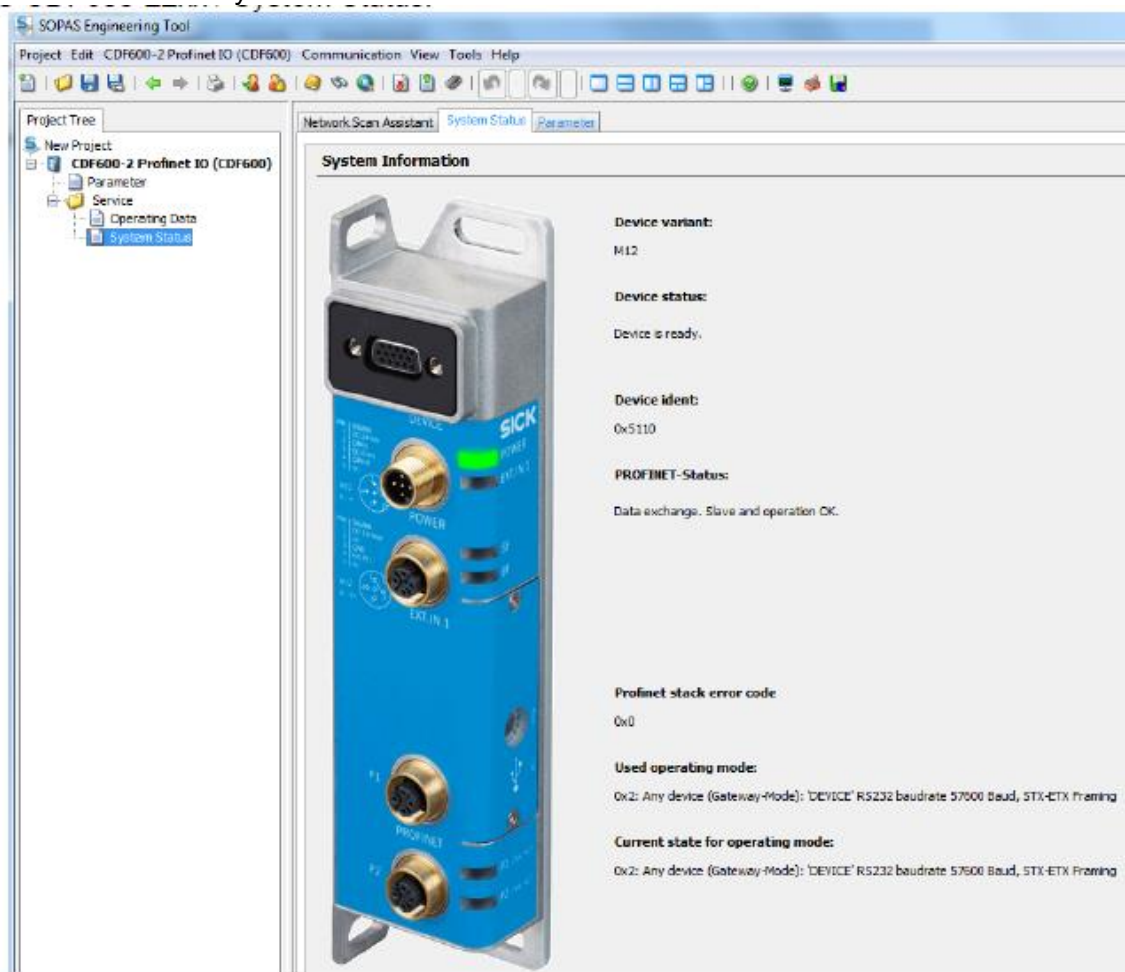
The switch setting and the PROFINET status can be viewed on the System Status tab for the CDF600-22xx.

**TIP:** In this mode, changes to the rotary switch are also directly displayed.

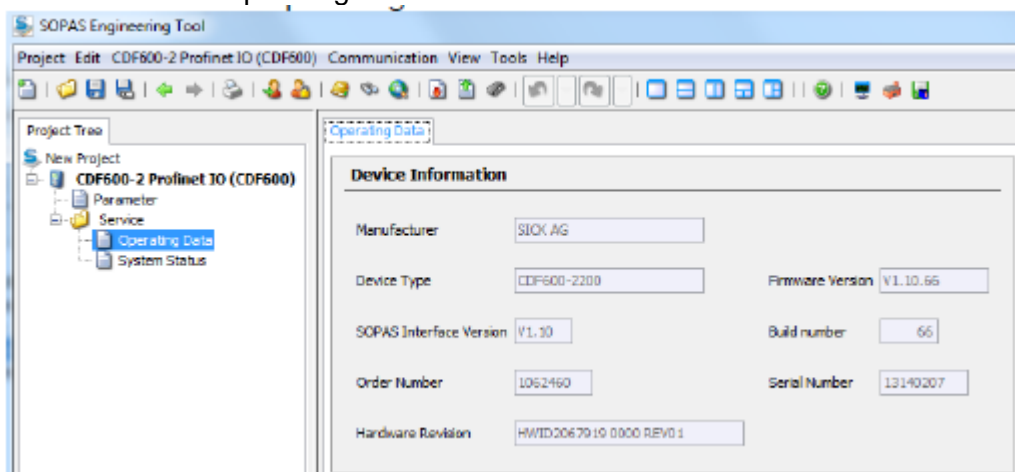
However, a change to the mode is only applied once the CDF600-22xx has been switched on/off.



## SOPAS CDF600-22xx / System Status:



## SOPAS CDF600-22xx / Operating Data:



In this mode, the connected ID sensor cannot be configured via the CDF600-22xx or the serial data displayed.

The ID sensor to be connected must be set previously in a suitable way. If the ID sensor has a direct USB or Ethernet interface, these can also be used for configuration.

## 5 Handling on the PLC side

### 5.1 Installing the GSDML file

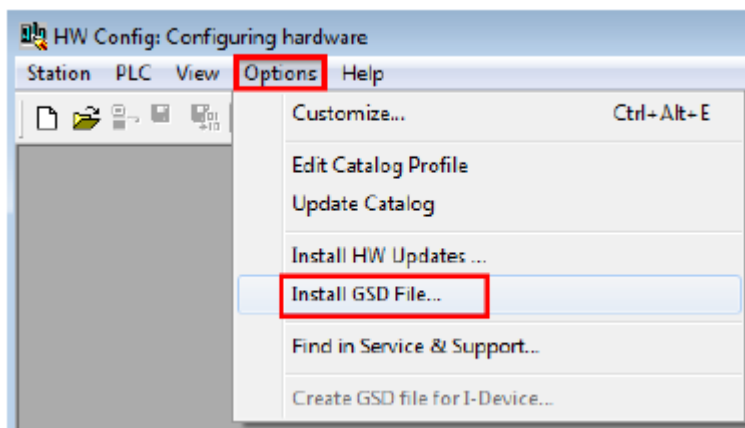
The current GSDML files for the relevant sensors to be connected can be downloaded from the following product page on the Internet:

[www.sick.com/software](http://www.sick.com/software), Select software type “GSD data file / GSDML”.

Install the GSDML file associated with the ID sensor to be connected as follows:

In the HW Config. Select “Extras > Install GSD files...”.

Then select “Options > Update Catalog”.



The selected device is installed and displayed in the catalog as follows:

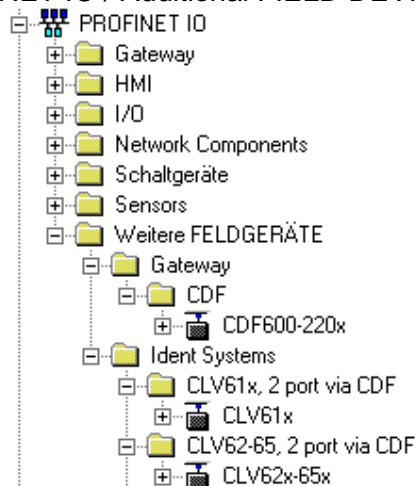
For 4Dpro devices that can be operated in the Proxy mode of the CDF600-22xx:

“PROFINET IO / Additional Field Devices / Ident Systems / CLV61x”

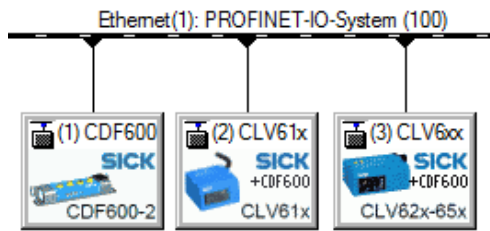
“PROFINET IO / Additional Field Devices / Ident Systems / CLV62x-65x”

For other serial devices connected in the Gateway mode of the CDF600-22xx:

“PROFINET IO / Additional FIELD DEVICES / Gateway / CDF / CDF600-22”

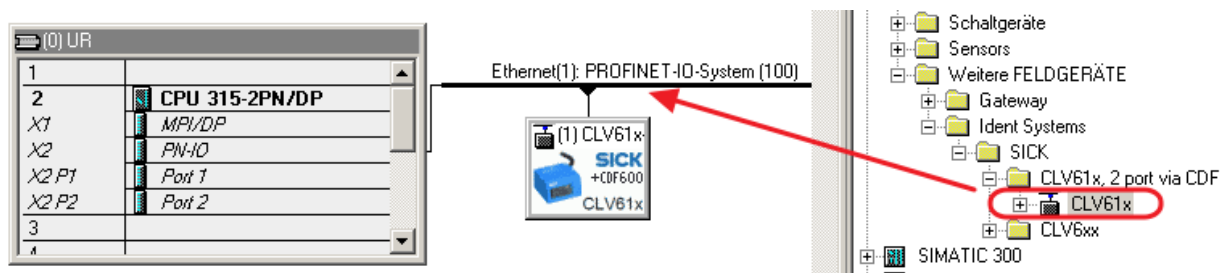


As shown in HW Config:

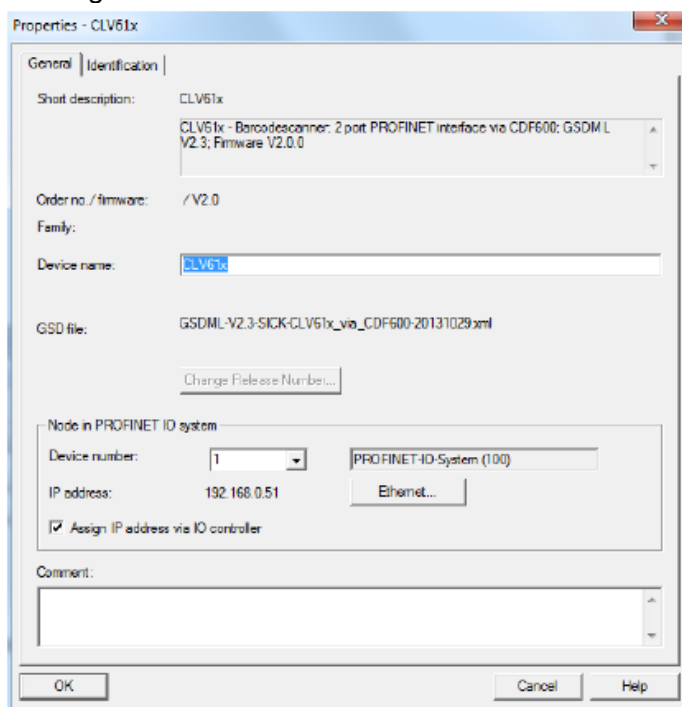


## 5.2 Adding sensors in HW Config

To add the ID sensors to be connected in the system, select the required device, e.g., “CLV61x”, and drag-and-drop to the PROFINET line.



You can select the **Device name**, for identifying the device in PROFINET by double clicking on the module. This name must **match** the name that you have given or will give to the CDF600-22xx via SOPAS-ET or via PLC HW Config.



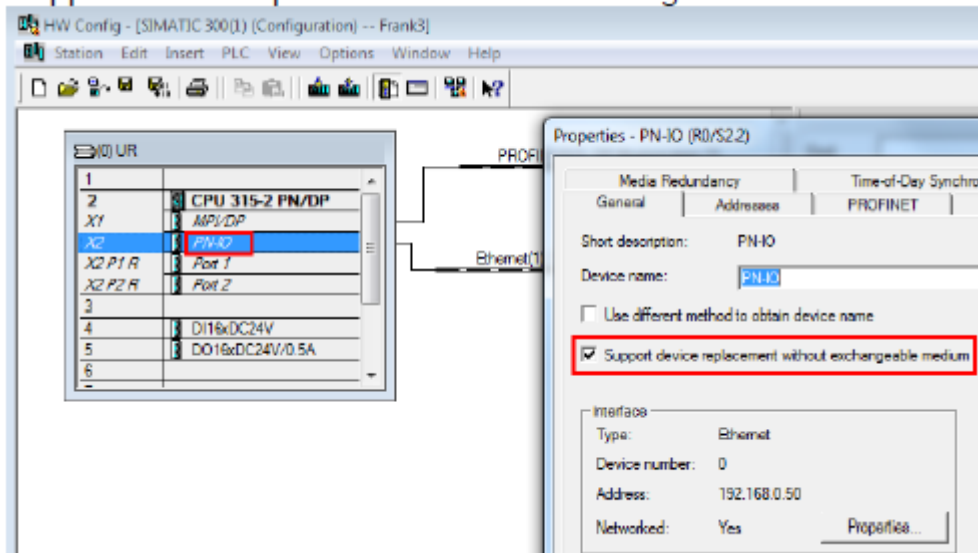
If you activate “Assign IP address via IO controller”, the PLC automatically assigns the IP address when a device with the corresponding PROFINET device name has been found.

Make sure that **no duplicate IP addresses** are used. If the IP sensor has its own Ethernet interface and this is also connected to the network, ensure that the CDF600-22xx and IP sensor have different IP addresses. If two devices have the same IP address, change one address before continuing. Use the SOPAS-ET Auto IP Scan, for example, to change the IP address of the sensor, or change the IP address of the CDF600-22xx via the PLC HW Config.

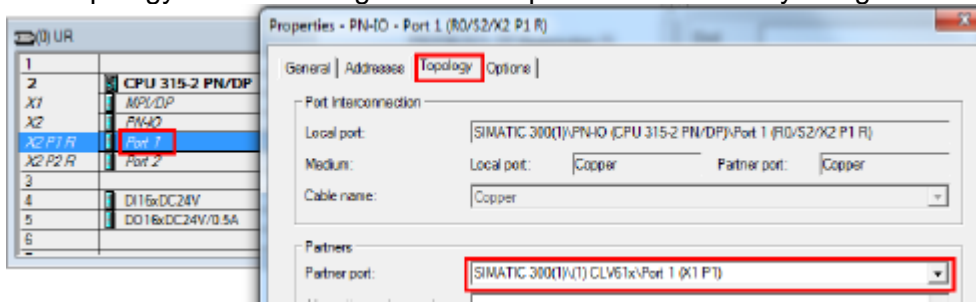
### 5.3 Assigning PN names automatically

The PN name can be automatically assigned by the PROFINET Controller (PLC), if:

- a) This has been activated in the CPU (is activated by default):  
Support device replacement without exchangeable medium = active



- b) The topology has been taught-in. A PN partner is then firmly assigned to each used port.



- c) The switches used must support this function, e.g., Scalance X200 series.

If a PN device is connected with an **empty name** and its type matches, the PN name stored on the PROFINET Controller (PLC) is assigned.

## 6 Data channel of the CDF600-22xx

Various communication protocols can be selected for the data channel in the CDF600-22xx. These modes are specific to SICK and require appropriate handling on the software side. The various communication protocols are described in the chapters below.

### 6.1 Handshake mode / Confirmed Messaging (Proxy/Gateway mode)

In **Proxy mode**, the data channel of the CDF600-22xx is operated in Handshake mode / Confirmed Messaging, when the communication protocol has been set to **CDF600 Handshake mode** (default setting). In **Gateway mode**, the data channel is always fixed to this mode.

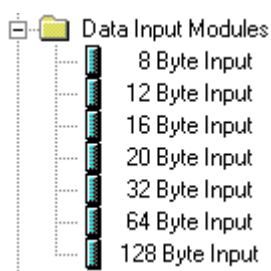
The data channel is compatible with the Byte Handshake mode of the CDF600 PROFIBUS, CMF400 PROFIBUS, CDM425, and PROFINET on Board. Send and receive with max. telegram length of 4000 bytes is possible with blocking.

In this mode, **one input and one output module** must be added in the HW Config for the ID sensor or CDF600-22xx. The size can be selected.

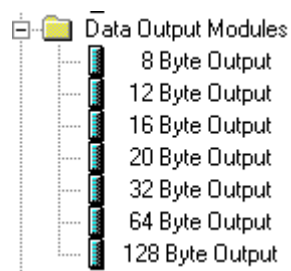
A data size of up to 128 bytes (= 64 words) can be selected.

In the input/output modules, 5 bytes are used for administration. A module may contain from e.g., 32 bytes to 27 bytes of user data without having to use blocking.

Input module (1 – 128 bytes)



Output module (1 – 128 bytes)



Standard: “32 byte input” and “32 byte output”

Adding one input and one output module as well as the Ctrl bits:

Slot	Module	Order number	I address	Q address	Diagnostic address	Comment
0	CLV61x				204	
X1	Interface				204	
X1 P1	Port 1				204	
X1 P2	Port 2				204	
1	Ctrl Bits in		0...1	0...1		
2	Ctrl Bits out					
3	32 Byte Input		256...287	256...287		
4	32 Byte Output					

**Note:** In the case of the CLV61x and CLV62x-65x via CDF600-22xx, the input and output modules are added when the device is added. These can be changed if required.

In addition, the I/O address must be selected.

Properties - 32 Byte Input - (R-/S3)

General | Addresses

Inputs

Start: 256 Process image: OB1 PI

End: 287

Properties - 32 Byte Output - (R-/S4)

General | Addresses

Outputs

Start: 256 Process image: OB1 PI

End: 287

The data must be handled **consistently**. Therefore, the I/O address for an S7-300/400 either must be assigned so that it lies **within its process image** or the **SFC 14/15** modules are used for consistent data transmission.

The size of the process image can be viewed by double-clicking on the CPU in the hardware configuration of S7. More details are available in the S7 documentation.

*Example:*

A process image size for the S7 CPU of 128 bytes is used here:

Properties - CPU 315-2 PN/DP - (RQ/S2)

Diagnostics/Clock | Protection | Communication

General | Startup | Synchronous Cycle

Cycle/Clock Memory | Retentive Memory | Interrupts | Time-of-Day Interrupts

Cycle

☒ Update OB1 process image cyclically

Scan cycle monitoring time [ms]: 150

Minimum scan cycle time [ms]: 0

Scan cycle load from communication [%]: 20

☐ Prioritized OCM communication

Size of the process-image input area: 512

Size of the process-image output area: 512

OB85 - call up at I/O access error: No OB85 call up

### 6.1.1 Using the SICK function block for PROFIBUS/PROFINET

If the optional SICK function block is used, the addresses of the I/O range incl. the correct length must be assigned to the block. The block then takes over the handling of the Handshake mode and can thus also receive data longer than the input range.

Download at [www.sick.com/software](http://www.sick.com/software) category "Function block".

If desired, optional actions such as triggering via command can also be used. More details are available in the documentation of the function block.

The function block for CLV6xx/Lector is shown below as an example:

The screenshot displays the SIMATIC Manager interface. On the left, the 'HW Config' window shows a rack configuration with a CPU 315-2 PN/DP and a SICK CLV61x module. The 'Properties' window for the CLV61x module shows the following I/O addresses:

Slot	Module	Order number	I address	Q address	Diagnostic
0	CLV61x				2043
X1	Interface				2042
X1.P1	Port 1				2041
X1.P2	Port 2				2040
1	Out Bits in		0..1	0..1	
2	Out Bits out			0..1	
3	32 Byte Input		256..287		
4	32 Byte Output			256..287	

The LAD/STL/FBD editor shows the SICK LECTOR CLV ENDP function block with the following parameters:

```

CALL "SICK LECTOR CLV ENDP", "INSTANCE_FB72"
IN_ADDR := W#16#100 // !!! Hex !!!
IN_LEN := -32 // decimal
OUT_ADDR := W#16#100 // !!! Hex !!!
OUT_LEN := -32 // decimal
CAN_ID := 0
TOUT := T#10S
START_REQ := L20.0
TRIG_ON := L20.1
TRIG_OFF := L20.2
MATCH_CODE := L20.3
COM_TEST := L20.4
SAVE_PERM := L20.5
FREE_COMMAND := L20.6
RESET := L20.7
DATA := "1_SICK LECTOR CLV DATA"
RD_DONE := "RD_Done"
REQ_DONE := "REQ_Done"
REQ_BUSY := "REQ_Busy"
ERROR := "Error"
ERRORCODE := "Errorcode"
NOP 0
    
```

### 6.1.2 Byte layout CDF600

The data storage area is set up in CDF600 mode as shown below.

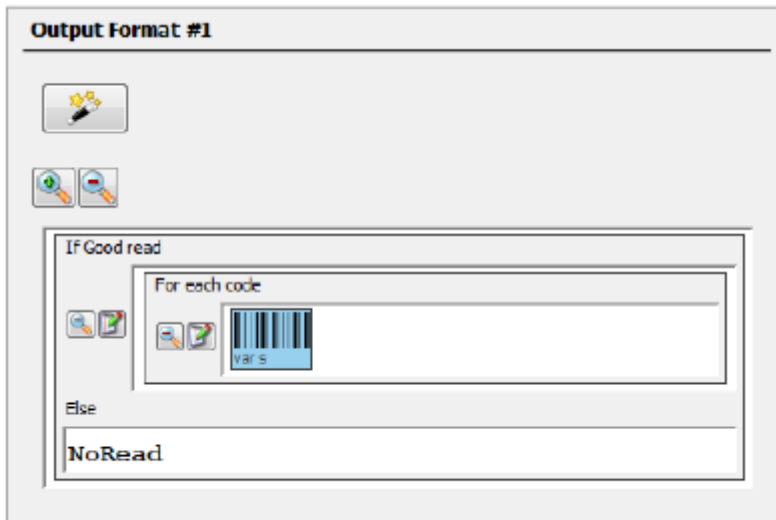
The length of the data storage area is up to 128 bytes depending on the module selected.

Address	Inputs (Data CLV → PLC)		Output (Data PLC → CLV)
1	Binary Status Bits In		Binary Status Bits Out
2	ReceiveCount (counter)	→	ReceiveCountBack (counter)
3	TransmitCountBack (counter)	←	TransmitCount (counter)
4	ReceiveLength Lowbyte		TransmitLength Lowbyte
5	ReceiveLength Highbyte		TransmitLength Highbyte
6	ReceiveData, Byte 1		TransmitData, Byte 1
7	ReceiveData, Byte 2		TransmitData, Byte 2
...	...		...
n	ReceiveData, Byte n - 5		TransmitData, Byte n - 5



### 6.1.3 Receiving data

Transmission from the ID sensor in Proxy mode of the CDF600-22xx requires no STX/ETX framing characters. If the output format contains STX/ETX, it is transmitted as data content to the PLC. The output format must be formatted **without STX/ETX** in the sensor:



The bus module puts the received data in the [ReceiveData](#) field and also adds the [ReceiveLength](#). The [ReceiveCount](#) value is also incremented to show that new data has been received.

#### Byte handshake:

To show that the PLC has correctly received the data, the PLC must respond to the CDF600-22xx by copying the [ReceiveCount](#) value to the output side for [ReceiveCountBack](#) within 10 seconds. The second input byte must be copied to the second output byte. (Byte handshake →). If the CDF600-22xx determines that both values are identical, it can send the next data block. The [ReceiveCount](#) runs from 1 to 255. 0 is omitted in normal operation.

If the CDF600-22xx sets the value to 0 during operation, it indicates that an error has occurred. In such case, the count must be restarted. To restart the count, the PLC must respond with 0. Otherwise, the count and data transmission does not continue. The PLC must always copy the [ReceiveCount](#) value to the [ReceiveCountBack](#) without restriction.

#### Longer data telegrams are divided into blocks (fragmentation):

If the received data is too long and does not fit in the input range of the PLC completely, it is automatically divided into blocks. The [ReceiveLength](#) always shows the length of the remaining data. In the first block, the length corresponds to the complete telegram length. If the block was answered with the byte handshake, the next block is sent and the length decremented.

Example:

Receipt of 100 bytes with a block size of 32-byte input (may contain up to 27 bytes in one block)

ReceiveCount	ReceiveLength	ReceiveData
1	100	first 27 data bytes
2	73	next 27 data bytes
3	46	next 27 data bytes
4	19	last 19 data bytes, the rest will be filled with 00

The [ReceiveLength](#) must be checked to determine whether a single block or the start of longer fragmented data was received. If [ReceiveLength](#) is longer than the data input variable, then fragmentation occurs. If shorter, it is a single telegram.



### 6.1.4 Example 1, receipt of a single block telegram (HS mode):

Input: 16 bytes, output: 16 bytes, data telegram with up to 11 bytes → single block

This can be done by triggering the ID sensor, e.g. via the hardware. Only data can then be received by the PLC.

Time:				1	2	3	4
// Input Byte - 16 Byte (Result Data received from CDM / Sensor)							
EB 50	"Stat-In-Bits"	BIN	2#0000_0100	2#0000_0100	2#0000_0100	2#0000_0000	2#0000_0000
EB 51	"Rec-Cnt"	DEC	1	1	2	2	2
EB 52	"Tr-Cnt-back"	DEC	0	0	0	0	0
EB 53	"Rec-Len-Low"	DEC	11	11	9	9	9
EB 54	"Rec-Len-High"	DEC	0	0	0	0	0
EB 55	"Rec-Data-1"	CHARACTER	'C'	'C'	'1'	'1'	'1'
EB 56	"Rec-Data-2"	CHARACTER	'L'	'L'	'2'	'2'	'2'
EB 57	"Rec-Data-3"	CHARACTER	'V'	'V'	'3'	'3'	'3'
EB 58	"Rec-Data-4"	CHARACTER	'6'	'6'	'4'	'4'	'4'
EB 59	"Rec-Data-5"	CHARACTER	'x'	'x'	'5'	'5'	'5'
EB 60	"Rec-Data-6"	CHARACTER	'x'	'x'	'6'	'6'	'6'
EB 61	"Rec-Data-7"	CHARACTER	'I'	'I'	'7'	'7'	'7'
EB 62	"Rec-Data-8"	CHARACTER	'D'	'D'	'8'	'8'	'8'
EB 63	"Rec-Data-9"	CHARACTER	'a'	'a'	'9'	'9'	'9'
EB 64	"Rec-Data-10"	CHARACTER	't'	't'	B#16#00	B#16#00	B#16#00
EB 65	"Rec-Data-11"	CHARACTER	'a'	'a'	B#16#00	B#16#00	B#16#00
// Output bytes - 16 Byte (e.g. Commands send to CDM / Sensor)							
AB 50	"Stat-Out-Bits"	BIN	2#0000_0000	2#0000_0000	2#0000_0000	2#0000_0000	2#0000_0000
AB 51	"Rec-Cnt-Back"	DEC	0	1	1	2	2
AB 52	"Tr-Cnt"	DEC	0	0	0	0	0
AB 53	"Tr-Len-Low"	DEC	0	0	0	0	0
AB 54	"Tr-Len-High"	DEC	0	0	0	0	0
AB 55	"Tr-Data-1"	CHARACTER	B#16#00	B#16#00	B#16#00	B#16#00	B#16#00
AB 56	"Tr-Data-2"	CHARACTER	B#16#00	B#16#00	B#16#00	B#16#00	B#16#00
AB 57	"Tr-Data-3"	CHARACTER	B#16#00	B#16#00	B#16#00	B#16#00	B#16#00
AB 58	"Tr-Data-4"	CHARACTER	B#16#00	B#16#00	B#16#00	B#16#00	B#16#00
AB 59	"Tr-Data-5"	CHARACTER	B#16#00	B#16#00	B#16#00	B#16#00	B#16#00
AB 60	"Tr-Data-6"	CHARACTER	B#16#00	B#16#00	B#16#00	B#16#00	B#16#00
AB 61	"Tr-Data-7"	CHARACTER	B#16#00	B#16#00	B#16#00	B#16#00	B#16#00
AB 62	"Tr-Data-8"	CHARACTER	B#16#00	B#16#00	B#16#00	B#16#00	B#16#00
AB 63	"Tr-Data-9"	CHARACTER	B#16#00	B#16#00	B#16#00	B#16#00	B#16#00
AB 64	"Tr-Data-10"	CHARACTER	B#16#00	B#16#00	B#16#00	B#16#00	B#16#00
AB 65	"Tr-Data-11"	CHARACTER	B#16#00	B#16#00	B#16#00	B#16#00	B#16#00

#### Remarks:

Time 1: The initial data block "CLV6xx-Data" (11 bytes) has been received and the PLC shows ReceiveCount = 1.

Time 2: The PLC has detected the data and confirms it by copying from ReceiveCount to ReceiveCountBack. The CDF600-22xx is now ready for the next data block.

Time 3: The next data block "123456789" (9 bytes) has been received by the CDF600-22xx and the PLC shows ReceiveCount = 2

Time 4: The PLC has detected the data and confirms it by copying from ReceiveCount to ReceiveCountBack. The CDF600-22xx is now ready for the next data block.

### 6.1.5 Example 2, receipt of a blocked telegram (HS mode):

Input: 16 bytes, output: 16 bytes, data telegram "CLV6xx-12345" with up to 12 bytes → divided into 2 blocks  
This can be done by triggering the ID sensor, e.g. via the hardware. Only data can then be received by the PLC.

				Time:	1	2/3	4
// Input Byte - 16 Byte (Result Data received from CDM / Sensor)							
EB 50	"Stat-In-Bits"	BIN	2#0000_0000		2#0000_0000	2#0000_0000	2#0000_0100
EB 51	"Rec-Cnt"	DEC	1		2		2
EB 52	"Tr-Cnt-back"	DEC	0		0		0
EB 53	"Rec-Len-Low"	DEC	12		1		1
EB 54	"Rec-Len-High"	DEC	0		0		0
EB 55	"Rec-Data-1"	CHARACTER	'C'		'S'		'S'
EB 56	"Rec-Data-2"	CHARACTER	'L'		B#16#00		B#16#00
EB 57	"Rec-Data-3"	CHARACTER	'V'		B#16#00		B#16#00
EB 58	"Rec-Data-4"	CHARACTER	'6'		B#16#00		B#16#00
EB 59	"Rec-Data-5"	CHARACTER	'x'		B#16#00		B#16#00
EB 60	"Rec-Data-6"	CHARACTER	'x'		B#16#00		B#16#00
EB 61	"Rec-Data-7"	CHARACTER	'I'		B#16#00		B#16#00
EB 62	"Rec-Data-8"	CHARACTER	'1'		B#16#00		B#16#00
EB 63	"Rec-Data-9"	CHARACTER	'2'		B#16#00		B#16#00
EB 64	"Rec-Data-10"	CHARACTER	'3'		B#16#00		B#16#00
EB 65	"Rec-Data-11"	CHARACTER	'4'		B#16#00		B#16#00
// Output bytes - 16 Byte (e.g. Commands send to CDM / Sensor)							
AB 50	"Stat-Out-Bits"	BIN	2#0000_0000		2#0000_0000	2#0000_0000	2#0000_0000
AB 51	"Rec-Cnt-Back"	DEC	0		1		2
AB 52	"Tr-Cnt"	DEC	0		0		0
AB 53	"Tr-Len-Low"	DEC	0		0		0
AB 54	"Tr-Len-High"	DEC	0		0		0
AB 55	"Tr-Data-1"	CHARACTER	B#16#00		B#16#00		B#16#00
AB 56	"Tr-Data-2"	CHARACTER	B#16#00		B#16#00		B#16#00
AB 57	"Tr-Data-3"	CHARACTER	B#16#00		B#16#00		B#16#00
AB 58	"Tr-Data-4"	CHARACTER	B#16#00		B#16#00		B#16#00
AB 59	"Tr-Data-5"	CHARACTER	B#16#00		B#16#00		B#16#00
AB 60	"Tr-Data-6"	CHARACTER	B#16#00		B#16#00		B#16#00
AB 61	"Tr-Data-7"	CHARACTER	B#16#00		B#16#00		B#16#00
AB 62	"Tr-Data-8"	CHARACTER	B#16#00		B#16#00		B#16#00
AB 63	"Tr-Data-9"	CHARACTER	B#16#00		B#16#00		B#16#00
AB 64	"Tr-Data-10"	CHARACTER	B#16#00		B#16#00		B#16#00
AB 65	"Tr-Data-11"	CHARACTER	B#16#00		B#16#00		B#16#00

#### Remarks:

- Time 1: The first data block "CLV6xx-1234" (the first 11 bytes) has been received and the PLC shows ReceiveCount = 1. The ReceiveLength is 12 bytes.
- Time 2: The PLC has detected the data and because it has noted that 12 bytes does not fit in the input range, it knows that additional blocks follow. It confirms the first block by copying ReceiveCount to ReceiveCountBack.
- Time 3: The CDF600-22xx sends the next block "5" (just 1 byte is left) to the PLC with ReceiveCount = 2 and ReceiveLength = 1.
- Time 4: The PLC has detected the next block and confirms it by copying from ReceiveCount to ReceiveCountBack. Transmission of the entire telegram has been completed.

The transportable data volume depends on the input range selected.

Input size minus 5 → Input block size

The max. length of a data telegram is 4000 bytes divided into several blocks.

## 6.1.6 Send data

The PLC can also send data, such as commands, to the CDF600-22xx or ID sensor. This is optional and only required if actions such as serial triggering must be carried out. In the Proxy mode of the CDF600-22xx, only commands that produce an echo or response, e.g., SOPAS commands, are permitted.

Data can be sent and received independently. It is recommended to first implement data receipt in the PLC and then to start creating the send routine.

To send data, it must be copied to the output range of the CDF600-22xx. The send data is entered without the STX/ETX frame.

Example:

PLC output range: 4 bytes with content: **s R I 0**

→ Sent as SOPAS command to ID sensor and answered with “sRA ...”.

The **Send begins by incrementing** the *TransmitCount* value by 1. The value must run from 1 to 255 and then start again at 1. The 0 must be omitted.

By copying the *TransmitCount* to *TransmitCountBack*, the CDF600-22xx confirms that the data has been sent. **Before the next data block can be sent**, the PLC must wait until these values are **again identical**.

If the CDF600-22xx has set the *TransmitCountBack* to 0, it indicates that an error has occurred and the count must start again. The PLC also has to set the *TransmitCount* to 0 for about 1 second to confirm that the error was detected. Data can then be sent again.

The cause of an error may be:

- Invalid length > 4000 bytes
- A transmission started with the first block, but subsequent blocks could not be sent or were faulty.
- *TransmitCount* was not incremented in the correct order.  
The counting must be 1, 2, 3, ... 255, 1, 2, 3, etc.

### Longer send data must be divided into blocks (fragmentation):

If a longer data telegram needs to be sent, it can be transmitted to the CDF600-22xx in blocks.

Therefore, start with the first block and always set the length to the data length to be sent. If the block was answered by the CDF600-22xx with *TransmitCountBack* as a byte handshake, the next block can be sent and the length decremented. The next block must follow within 10 seconds. Otherwise, a timeout occurs.

Example:

Transmission of 100 bytes with a block size of 32-byte output (may contain up to 27 bytes in one block)

TransmitCount	TransmitLength	TransmitData
1	100	first 27 data bytes
2	73	next 27 data bytes
3	46	next 27 data bytes
4	19	last 19 data bytes

### 6.1.7 Example 3, transmission of a single block telegram (HS mode)

Input: 16 bytes, output: 16 bytes, data telegram with 4 bytes "sRI0" is sent.

The ID sensor responds with "sRA ...".

The command can be used to query the software version of the sensor / SDD file or to generally check the communication channel. This command can be used for all 4Dpro sensors (CLV6xx, Lector62x, RFH6xx, RFU6xx, etc.) irrespective of the configuration.

			Time:	1/2	3/4	5	6/7	8
// Input Data - 16 byte (result Data received from PLC)								
EB 50	"Stat-In-Bits"	BIN		2#0000_0000	2#0000_0000	2#0000_0100	2#0000_0000	2#0000_0100
EB 51	"Rec-Cnt"	DEZ		1	2	2	3	3
EB 52	"Tr-Cnt-Back"	DEZ		1	1	1	2	2
EB 53	"Rec-Len-Low"	DEZ		22	11	11	6	6
EB 54	"Rec-Len-High"	DEZ		0	0	0	0	0
EB 55	"Rec-Data-1"	ZEICHEN		's'	'6'	'6'	's'	's'
EB 56	"Rec-Data-2"	ZEICHEN		'R'	'2'	'2'	'F'	'F'
EB 57	"Rec-Data-3"	ZEICHEN		'A'	'x'	'x'	'A'	'A'
EB 58	"Rec-Data-4"	ZEICHEN		' '	' '	' '	' '	' '
EB 59	"Rec-Data-5"	ZEICHEN		'0'	'5'	'5'	'1'	'1'
EB 60	"Rec-Data-6"	ZEICHEN		' '	' '	' '	'1'	'1'
EB 61	"Rec-Data-7"	ZEICHEN		'6'	'v'	'v'	B#16#00	B#16#00
EB 62	"Rec-Data-8"	ZEICHEN		' '	'5'	'5'	B#16#00	B#16#00
EB 63	"Rec-Data-9"	ZEICHEN		'C'	' '	' '	B#16#00	B#16#00
EB 64	"Rec-Data-10"	ZEICHEN		'L'	'1'	'1'	B#16#00	B#16#00
EB 65	"Rec-Data-11"	ZEICHEN		'V'	'1'	'1'	B#16#00	B#16#00
// Output Data - 16 byte (e.g. Commands send from PLC)								
AB 50	"Stat-Out-Bits"	BIN		2#0000_0000	2#0000_0000	2#0000_0000	2#0000_0000	2#0000_0000
AB 51	"Rec-Cnt-Back"	DEZ		0	1	2	2	3
AB 52	"Tr-Cnt"	DEZ		1	1	1	2	2
AB 53	"Tr-Len-Low"	DEZ		4	4	4	4	4
AB 54	"Tr-Len-High"	DEZ		0	0	0	0	0
AB 55	"Tr-Data-1"	ZEICHEN		's'	's'	's'	's'	's'
AB 56	"Tr-Data-2"	ZEICHEN		'R'	'R'	'R'	'R'	'R'
AB 57	"Tr-Data-3"	ZEICHEN		'I'	'I'	'I'	'I'	'I'
AB 58	"Tr-Data-4"	ZEICHEN		'0'	'0'	'0'	'X'	'X'
AB 59	"Tr-Data-5"	ZEICHEN		B#16#00	B#16#00	B#16#00	B#16#00	B#16#00
AB 60	"Tr-Data-6"	ZEICHEN		B#16#00	B#16#00	B#16#00	B#16#00	B#16#00
AB 61	"Tr-Data-7"	ZEICHEN		B#16#00	B#16#00	B#16#00	B#16#00	B#16#00
AB 62	"Tr-Data-8"	ZEICHEN		B#16#00	B#16#00	B#16#00	B#16#00	B#16#00
AB 63	"Tr-Data-9"	ZEICHEN		B#16#00	B#16#00	B#16#00	B#16#00	B#16#00
AB 64	"Tr-Data-10"	ZEICHEN		B#16#00	B#16#00	B#16#00	B#16#00	B#16#00
AB 65	"Tr-Data-11"	ZEICHEN		B#16#00	B#16#00	B#16#00	B#16#00	B#16#00

#### Remarks:

- Time 1: First data telegram "sRI0" (SDD version) was sent by setting TransmitCount = 1 on the output side. If the CDF600-22xx has sent the data, it confirms it by copying TransmitCount to TransmitCountBack on the input side.
- Time 2: The ID sensor responds with "sRA 0 6 CLV62x 5 V5.11" (22 bytes). The first 11 bytes ("sRA 0 6 CLV") are reported to the PLC with ReceiveCount = 1 and ReceiveLength 22 bytes (total length). The PLC has detected the data and because it has noted that 22 bytes does not fit in the input range, it knows that another block follows. It confirms the first block by copying ReceiveCount to ReceiveCountBack.
- Time 3: The CDF600-22xx immediately sends the next (last) block "62x 5 V5.11" (11 bytes) to the PLC with ReceiveCount = 2 and ReceiveLength = 11.

- Time 4/5: The PLC has detected the additional data and confirms it by copying from ReceiveCount to ReceiveCountBack. The CDF600-22xx is now ready for the next data block.
- Time 6: The next data telegram “sRIX” (here a faulty command) was sent by incrementing TransmitCount on the output side to 2. The CDF600-22xx confirms this by TransmitCountBack = 2 on the input side.
- Time 7: The ID sensor responds here with an error message “sFA 11” (6 bytes, FA = error response, 11 = character error). This is reported to the PLC with ReceiveCount = 3 and ReceiveLength 6 bytes.
- Time 8: The PLC has detected the data and confirms it by copying the ReceiveCount to ReceiveCountBack. The CDF600-22xx is now ready for additional data.

It is also advisable to test the required commands in advance e.g., using the SOPAS terminals connected to the serial AUX interface or the AUX port of the Ethernet interface of the ID sensor. The ID sensor can process SOPAS commands from any of its data interfaces.

### 6.1.8 Example 4, transmission of a blocked telegram (HS mode):

Input: 16 bytes, output: 16 bytes, data telegram software trigger “sMN mTCgateon” with 13 bytes.

Because the telegram is too long, 2 blocks are required.

This command can be used for all 4Dpro sensors (CLV6xx, Lector6xx, RFH6xx, RFU6xx, etc.) to trigger. The trigger source must be configured to (SOPAS) command.

A successful reading with output is immediately shown. If desired, the command “sMN mTCgateoff” (14 bytes) can also be used to terminate the trigger. This is not shown here.

			Time: 1	2/3	4	5	6	7
// Input Data - 16 byte (result Data received)								
EB 50	"Stat-In-Bits"	BIN	2#0000_0000	2#0000_0100	2#0000_0000	2#0000_0100	2#0000_0100	2#0000_0100
EB 51	"Rec-Cnt"	DEZ	0	1	2	2	3	3
EB 52	"Tr-Cnt-Back"	DEZ	1	2	2	2	2	2
EB 53	"Rec-Len-Low"	DEZ	0	15	4	4	10	10
EB 54	"Rec-Len-High"	DEZ	0	0	0	0	0	0
EB 55	"Rec-Data-1"	ZEICHEN	B#16#00	's'	'o'	'o'	'1'	'1'
EB 56	"Rec-Data-2"	ZEICHEN	B#16#00	'A'	'n'	'n'	'2'	'2'
EB 57	"Rec-Data-3"	ZEICHEN	B#16#00	'N'	''	''	'3'	'3'
EB 58	"Rec-Data-4"	ZEICHEN	B#16#00	''	'1'	'1'	'4'	'4'
EB 59	"Rec-Data-5"	ZEICHEN	B#16#00	'm'	B#16#00	B#16#00	'5'	'5'
EB 60	"Rec-Data-6"	ZEICHEN	B#16#00	'T'	B#16#00	B#16#00	'6'	'6'
EB 61	"Rec-Data-7"	ZEICHEN	B#16#00	'C'	B#16#00	B#16#00	'7'	'7'
EB 62	"Rec-Data-8"	ZEICHEN	B#16#00	'g'	B#16#00	B#16#00	'8'	'8'
EB 63	"Rec-Data-9"	ZEICHEN	B#16#00	'a'	B#16#00	B#16#00	'9'	'9'
EB 64	"Rec-Data-10"	ZEICHEN	B#16#00	't'	B#16#00	B#16#00	'0'	'0'
EB 65	"Rec-Data-11"	ZEICHEN	B#16#00	'e'	B#16#00	B#16#00	B#16#00	B#16#00
// Output Data - 16 byte (e.g. Commands)								
AB 50	"Stat-Out-Bits"	BIN	2#0000_0000	2#0000_0000	2#0000_0000	2#0000_0000	2#0000_0000	2#0000_0000
AB 51	"Rec-Cnt-Back"	DEZ	0	0	1	2	2	3
AB 52	"Tr-Cnt"	DEZ	1	2	2	2	2	2
AB 53	"Tr-Len-Low"	DEZ	13	2	2	2	2	2
AB 54	"Tr-Len-High"	DEZ	0	0	0	0	0	0
AB 55	"Tr-Data-1"	ZEICHEN	's'	'o'	'o'	'o'	'o'	'o'
AB 56	"Tr-Data-2"	ZEICHEN	'M'	'n'	'n'	'n'	'n'	'n'
AB 57	"Tr-Data-3"	ZEICHEN	'N'	'N'	'N'	'N'	'N'	'N'
AB 58	"Tr-Data-4"	ZEICHEN	''	''	''	''	''	''
AB 59	"Tr-Data-5"	ZEICHEN	'm'	'm'	'm'	'm'	'm'	'm'
AB 60	"Tr-Data-6"	ZEICHEN	'T'	'T'	'T'	'T'	'T'	'T'
AB 61	"Tr-Data-7"	ZEICHEN	'C'	'C'	'C'	'C'	'C'	'C'
AB 62	"Tr-Data-8"	ZEICHEN	'g'	'g'	'g'	'g'	'g'	'g'
AB 63	"Tr-Data-9"	ZEICHEN	'a'	'a'	'a'	'a'	'a'	'a'
AB 64	"Tr-Data-10"	ZEICHEN	't'	't'	't'	't'	't'	't'
AB 65	"Tr-Data-11"	ZEICHEN	'e'	'e'	'e'	'e'	'e'	'e'

#### Remarks:

- Time 1: The first data block of 11 bytes has been sent by setting TransmitCount = 1 with TransmitchLength = 13.  
If the CDF600-22xx has received the first data block, it confirms it by copying from TransmitCount to TransmitCountBack on the input side.
- Time 2: The second data block of the last 2 bytes has been sent by incrementing from TransmitCount = 2 with TransmitchLength = 2.  
When the CDF600-22xx has received the second data block, it confirms it by copying from TransmitCount to TransmitCountBack on the input side.
- Time 3: The ID sensor sends the echo of the trigger command “sMN mTCgateon” to the PLC with ReceiveCount = 1 and ReceiveLength 15 bytes.
- Time 4: The PLC confirms receipt of the first 11 bytes with ReceiveCountBack = 1.  
The ID sensor then sends the remaining 2 bytes to the PLC with ReceiveCount = 2 and RecLength = 2.
- Time 5: The PLC confirms receipt of the data with ReceiveCountBack = 2.
- Time 6: With “Output immediately”, the ID sensor sends the result immediately e.g. upon detection of the bar code. 10 bytes (“1234567890”) are reported to the PLC with ReceiveCount = 3 and ReceiveLength 10 bytes.

Time 7: The PLC confirms receipt of the data with ReceiveCountBack = 3.

The TransmitCount must begin again with 1 when 255 is reached, i.e., 0 must be omitted.

The transportable data volume depends on the output range selected.

Output size minus 5 → Output block size

The max. length of a data telegram is 4000 bytes divided into several blocks.

### 6.1.9 Binary status bits In

The first input byte (see chapter 6.1.2) displays the binary status bits In.

It contains some status bits and one heartbeat bit.

Bit	Name	Meaning
D7	–	Reserved
D6	<i>Transmit Buffer Overrun</i>	0: No error 1: The CDF600-22xx has received send data from the PLC and the queue is full. Reset: With the next successful transmit telegram sent from the PLC.
D5	<i>Receive Buffer Overrun</i>	0: No error 1: The receive buffer is full and the received data must be discarded. Reset: With the next modified ReceiveCountBack from the PLC side.
D4	–	Reserved
D3	<i>PLC Error</i>	0: No error 1: The CDF600-22xx identified a handling error when sending data in the fieldbus master (PLC). The fieldbus module does not accept the transmitted data if this error occurs and requires resynchronization with the fieldbus master. The PLC program must be corrected based on the error.  Possible causes: - Impermissible length (e.g., length > 4000). - The subsequent block was not received within 10 seconds. - TransmitCount was incremented in the wrong sequence.  If “PLC-Error” bit is set, no send data is transmitted. However, the receiver side is further processed. If “PLC-Error” bit is set, TransmitCountBack is set to 0.  The PLC Error bit is cleared after TransmitCount is reset to 0 by the PLC.
D2	<i>Heartbeat</i>	0/1: Changes every second – shows the presence of the CDF600-22xx
D1	–	Reserved
D0	–	Reserved

“Binary Status Bits” - Assignment in Handshake / No-Handshake mode



### 6.1.10 Binary Status Bits Out

The first output byte in the header is reserved. It must be set to zero.

Bit	Name	Meaning
D7	–	Reserved
D6	–	Reserved
D5	–	Reserved
D4	–	Reserved
D3	–	Reserved
D2	–	Reserved
D1	–	Reserved
D0	–	Reserved

“Binary Status Bits Out” - Assignment in Handshake / No-Handshake mode

## 6.2 No-Handshake Mode

In **Proxy mode**, the data channel of the CDF600-22xx is operated in No-Handshake mode, when the communication protocol has been set to **CDF600 No-Handshake**.

In **Gateway mode**, the data channel of the CDF600-22xx is operated in No-Handshake mode, when via SOPAS the communication protocol has been configured to **CDF600 No-Handshake**.

The data channel is compatible with the No-Handshake mode of the CDF600 PROFIBUS, CMF400 PROFIBUS, CDM425, and PROFINET on Board. The max. telegram length is limited by the size of the input/output range and is max. 123 bytes. There is no fragmenting / blocking.

In this mode, **one input and one output module, Ctrl bits In and Ctrl bits Out** must be added in HW Config for the ID sensor or CDF600-22xx. The size can be freely selected. In the input/output modules, 5 bytes are used for administration. A module may contain from e.g. 32 bytes to 27 bytes of user data.

This mode is only recommended if the length of the received data does not exceed the input range minus 5. It is also necessary to ensure that the PLC always has enough time to receive the data before the next data block arrives. Therefore, this mode is only recommended for testing or for individual devices with low communication load.

To select No-Handshake mode, “CDF600 No-Handshake” must be configured and permanently saved in the ID sensor. The mode is only active after restarting the CDF600-22xx with the attached ID sensor.

For the data channel, one input module and one output module must be selected as in CDF600-22xx Handshake mode. In terms of the layout, the mode is identical to the HS mode except that fragmentation is not possible on both the receiver and transmission sides.

The binary inputs and outputs are identical to the Handshake mode.  
The Ctrl bits can also be used.



## 6.2.1 Byte layout in No-Handshake Mode (NH):

Address	Inputs (Data CLV → PLC)		Output (Data PLC → CLV)
1	Binary inputs		Binary outputs
2	ReceiveCount (counter)	→	ReceiveCountBack (counter)
3	TransmitCountBack (counter)	←	TransmitCount (counter)
4	ReceiveLength Lowbyte		TransmitLength Lowbyte
5	ReceiveLength Highbyte		TransmitLength Highbyte
6	ReceiveData, Byte 1		TransmitData, Byte 1
7	ReceiveData, Byte 2		TransmitData, Byte 2
...	...		...
N	ReceiveData, Byte n - 5		TransmitData, Byte n - 5

The length of the data storage area is up to max. 123 bytes depending on the module selected. The first 5 bytes are used for administration and have special meaning. The remaining bytes from byte 6 onward are the ASCII data for sending / receiving.

The binary inputs and outputs are identical to the Handshake mode.

## 6.2.2 Receiving data in No-Handshake Mode

The data format is set as in Handshake mode without the STX/ETX frame.

The CDF600-22xx puts the received data in the *ReceiveData* field and also adds its *ReceiveLength*. The *ReceiveCount* value is also incremented to show that new data has been received.

If the data is too long, the overhanging data is truncated and lost during transmission.

If additional data is received, it is always presented to the PLC directly. It is not checked if the user program in the PLC has already received the data. As a result, data may be overwritten without warning. However, the user program can check whether the *ReceiveCount* value is incremented by more than 1 to determine that data has been overwritten.

### 6.2.3 Example 5, receive telegram (NH mode):

Input: 16 bytes, output: 16 byte, data telegrams with up to 11 bytes are possible. No handshake is required.  
This can occur, for example, by triggering the ID sensor via hardware, so that the PLC only has to realize data receipt.

Time:				1	2	3
// Input Byte - 16 Byte (Result Data received from CDM / Sensor)						
EB 50	"Stat-In-Bits"	BIN	2#0000_0100	2#0000_0000	2#0000_0100	
EB 51	"Rec-Cnt"	DEC	1	2	3	
EB 52	"Tr-Cnt-back"	DEC	0	0	0	
EB 53	"Rec-Len-Low"	DEC	8	4	6	
EB 54	"Rec-Len-High"	DEC	0	0	0	
EB 55	"Rec-Data-1"	CHARACTER	'1'	'S'	'N'	
EB 56	"Rec-Data-2"	CHARACTER	'2'	'I'	'o'	
EB 57	"Rec-Data-3"	CHARACTER	'3'	'C'	'R'	
EB 58	"Rec-Data-4"	CHARACTER	'4'	'K'	'e'	
EB 59	"Rec-Data-5"	CHARACTER	'5'	B#16#00	'a'	
EB 60	"Rec-Data-6"	CHARACTER	'6'	B#16#00	'd'	
EB 61	"Rec-Data-7"	CHARACTER	'7'	B#16#00	B#16#00	
EB 62	"Rec-Data-8"	CHARACTER	'8'	B#16#00	B#16#00	
EB 63	"Rec-Data-9"	CHARACTER	B#16#00	B#16#00	B#16#00	
EB 64	"Rec-Data-10"	CHARACTER	B#16#00	B#16#00	B#16#00	
EB 65	"Rec-Data-11"	CHARACTER	B#16#00	B#16#00	B#16#00	
// Output bytes - 16 Byte (e.g. Commands send to CDM / Sensor)						
AB 50	"Stat-Out-Bits"	BIN	2#0000_0000	2#0000_0000	2#0000_0000	
AB 51	"Rec-Cnt-Back"	DEC	0	0	0	
AB 52	"Tr-Cnt"	DEC	0	0	0	
AB 53	"Tr-Len-Low"	DEC	0	0	0	
AB 54	"Tr-Len-High"	DEC	0	0	0	
AB 55	"Tr-Data-1"	CHARACTER	B#16#00	B#16#00	B#16#00	
AB 56	"Tr-Data-2"	CHARACTER	B#16#00	B#16#00	B#16#00	
AB 57	"Tr-Data-3"	CHARACTER	B#16#00	B#16#00	B#16#00	
AB 58	"Tr-Data-4"	CHARACTER	B#16#00	B#16#00	B#16#00	
AB 59	"Tr-Data-5"	CHARACTER	B#16#00	B#16#00	B#16#00	
AB 60	"Tr-Data-6"	CHARACTER	B#16#00	B#16#00	B#16#00	
AB 61	"Tr-Data-7"	CHARACTER	B#16#00	B#16#00	B#16#00	
AB 62	"Tr-Data-8"	CHARACTER	B#16#00	B#16#00	B#16#00	
AB 63	"Tr-Data-9"	CHARACTER	B#16#00	B#16#00	B#16#00	
AB 64	"Tr-Data-10"	CHARACTER	B#16#00	B#16#00	B#16#00	
AB 65	"Tr-Data-11"	CHARACTER	B#16#00	B#16#00	B#16#00	

Remarks:

- Time 1: The first data block "12345678" (8 bytes) has been received and the PLC shows ReceiveCount = 1.
- Time 2: The second data block "SICK" (4 bytes) has been received and the PLC shows ReceiveCount = 2.
- Time 3: The third data block "NoRead" (6 bytes) has been received and the PLC shows ReceiveCount = 3.

## 6.2.4 Example 6, send telegram (NH mode)

Input: 16 bytes, output: 16 bytes, data telegram with 4 bytes "sRI0" is sent.

The ID sensor responds with "sRA ...".

The command can be used to query the software version of the ID sensor / SDD file or to generally check the communication channel. This command can be used for all 4Dpro sensors (CLV6xx, Lector62x, RFH6xx, RFU6xx, etc.) irrespective of the configuration.

			Time:	1/2	3/4
// Input Data - 16 byte (result Data received from PLC)					
EB 50	"Stat-In-Bits"	BIN		2#0000_0000	2#0000_0100
EB 51	"Rec-Cnt"	DEZ		1	2
EB 52	"Tr-Cnt-Back"	DEZ		1	2
EB 53	"Rec-Len-Low"	DEZ		22	6
EB 54	"Rec-Len-High"	DEZ		0	0
EB 55	"Rec-Data-1"	ZEICHEN		's'	's'
EB 56	"Rec-Data-2"	ZEICHEN		'R'	'F'
EB 57	"Rec-Data-3"	ZEICHEN		'A'	'A'
EB 58	"Rec-Data-4"	ZEICHEN		' '	' '
EB 59	"Rec-Data-5"	ZEICHEN		'0'	'1'
EB 60	"Rec-Data-6"	ZEICHEN		' '	'1'
EB 61	"Rec-Data-7"	ZEICHEN		'6'	B#16#00
EB 62	"Rec-Data-8"	ZEICHEN		' '	B#16#00
EB 63	"Rec-Data-9"	ZEICHEN		'C'	B#16#00
EB 64	"Rec-Data-10"	ZEICHEN		'L'	B#16#00
EB 65	"Rec-Data-11"	ZEICHEN		'V'	B#16#00
// Output Data - 16 byte (e.g. Commands send from PLC)					
AB 50	"Stat-Out-Bits"	BIN		2#0000_0000	2#0000_0000
AB 51	"Rec-Cnt-Back"	DEZ		0	0
AB 52	"Tr-Cnt"	DEZ		1	2
AB 53	"Tr-Len-Low"	DEZ		4	4
AB 54	"Tr-Len-High"	DEZ		0	0
AB 55	"Tr-Data-1"	ZEICHEN		's'	's'
AB 56	"Tr-Data-2"	ZEICHEN		'R'	'R'
AB 57	"Tr-Data-3"	ZEICHEN		'I'	'I'
AB 58	"Tr-Data-4"	ZEICHEN		'0'	'X'
AB 59	"Tr-Data-5"	ZEICHEN		B#16#00	B#16#00
AB 60	"Tr-Data-6"	ZEICHEN		B#16#00	B#16#00
AB 61	"Tr-Data-7"	ZEICHEN		B#16#00	B#16#00
AB 62	"Tr-Data-8"	ZEICHEN		B#16#00	B#16#00
AB 63	"Tr-Data-9"	ZEICHEN		B#16#00	B#16#00
AB 64	"Tr-Data-10"	ZEICHEN		B#16#00	B#16#00
AB 65	"Tr-Data-11"	ZEICHEN		B#16#00	B#16#00

Remarks:

- Time 1: First data telegram "sRI0" (SDD version) was sent by setting TransmitCount = 1 on the output side. If the CDF600-22xx has sent the data, it confirms it by copying TransmitCount to TransmitCountBack on the input side.
- Time 2: The ID sensor responds with "sRA 0 6 CLV62x 5 V5.11" (22 bytes). The first 11 bytes ("sRA 0 6 CLV") are reported to the PLC with ReceiveCount = 1 and ReceiveLength 22 bytes (total length). The remaining 11 bytes of data are lost. You may want to select a larger input range.
- Time 3: The next data telegram "sRIX" (here a faulty command) was sent by incrementing TransmitCount on the output side to 2. The CDF600-22xx confirms this by TransmitCountBack = 2 on the input side.

Time 4: The ID sensor responds here with an error message “sFA 11” (6 bytes, FA = error response, 11 = character error). This is reported to the PLC with ReceiveCount = 2 and ReceiveLength 6 bytes.

The TransmitCount must begin again with 1 when 255 is reached, i.e., 0 must be omitted.  
Before sending again, TransmitCount and TransmitCountBack must be identical.

It is also advisable to test the required commands in advance e.g., using the SOPAS terminals connected to the serial AUX interface or the AUX port of the Ethernet interface of the ID sensor. The ID sensor can process SOPAS commands from any of its data interfaces.

## 7 Function of the CDF600-22xx in Gateway mode

This mode is active if the rotary coding switch “Mode” of the CDF600-22xx is set to 2 or 4, and then the CDF600-22xx is restarted.

In this mode, the **GSDML file of the CDF600-22xx** (GSDML-V2.3-SICK-CDF600-20130602.xml or newer) must be used, an input and output module must be selected for the data channel, and the Ctrl bits in and Ctrl bits out must be integrated as in the Gateway mode of the CDF600-22xx.

Mode 2: Serial interface is fixed on 57.6 kBd, 8, n, 1

Mode 4: Serial interface is fixed on 9.6 kBd, 8, n, 1

In Gateway mode, the following features apply:

- For the PLC, the transmission mode **CDF600 (with handshake)** is standard for the data channel. This can only be changed to No-Handshake via SOPAS.
- The data on the serial interface must be **framed** with the control characters **STX / ETX**, if in the CDF600-2 the standard set serial protocol “Cola A” is used. The serial protocol can be changed via SOPAS to “Cola B” (binary) that works with length specifications. See also Chapter 4.3. In further documentation it is assumed that Cola A, i.e., the standard STX/ETX frame will be used.

### Receive data:

The data on the serial line must have STX at the beginning and ETX at the end.

These framing characters are **removed**, not displayed in the data field for the PLC and not counted in the length:

Serial line: **STX** **a b c** **ETX** → PLC input range: 3 bytes with content **a b c**

### Send data:

The data must be sent from the PLC without the STX/ETX frame and the framing characters are also not counted in the length.

The CDF600-22xx adds the framing characters to the serial line independently. This **cannot** be suppressed.

*Example:*

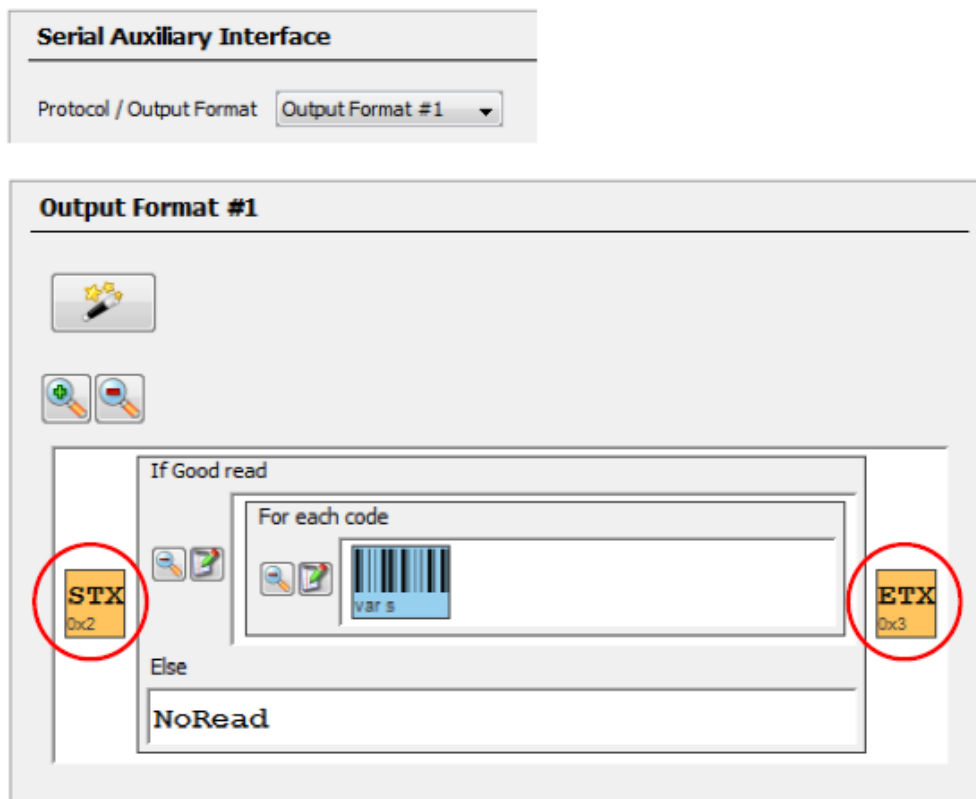
PLC output range: 4 bytes with content: **s R I 0** → Serial line: **STX** **s R I 0** **ETX**

### Connection of the ID sensor:

The serial interface is on pins 2, 3 and 5 in the CDF600-22xx. If an 4Dpro sensor is connected, the connection must be made using the serial **AUX interface** of the ID sensor and not using its host interface. The serial AUX interface may need to be assigned the appropriate **output format** and the output configured **with STX / ETX**.

*Example:*

Output format 1 with STX/ETX:



As digital I/O bits only the “EXT. IN 1” input is available in the least significant bit of the first byte as an input. See chapter 8.1.1.

## 8 Digital inputs/outputs

The PLC can read and control the information for the digital inputs/outputs of the ID sensor or CDF600-22xx. This depends on the mode of communication selected.

### 8.1 Ctrl bits

The Ctrl bits can be added in the HW Config before or after the data channel.

Slot	Module	Order number	I address	Q address	Diagnostic	C...
0	station51				200...	
X1	MPI/DP				200...	
X2	MPI/DP				200...	
X2 P1 R	Port 1				200...	
X2 P2 R	Port 2				200...	
1	Ctrl Bits in		0...1			
2	Ctrl Bits out			0...1		
3	32 Byte Input		256...287			
4	32 Byte Output			256...287		

### 8.1.1 Ctrl bits In

By adding the Ctrl bits In, the PLC can monitor the digital inputs and outputs of the ID sensor. Assignment of the Ctrl bits In can vary depending on the ID sensor. Assignment with bar code scanner CLV62x ... 65x is used here as an example.

Add.	Bit	Name	Meaning	Proxy mode	Gateway mode
Add.+1	D0	Device Ready	Status of the attached ID sensor or CDF600-22xx	DevReady of the ID sensors	DevReady of the CDF600
Add.+1	D1	System Ready **	The connected ID sensor is OK (device ready) and the monitored CAN devices are accessible (display in the net monitoring system status OK). **	X	---
Add.+1	D2	Good Read *	Status of the last read results *	X	---
Add.+1	D3	No Read *	Status of the last read results *	X	---
Add.+1	D4	External Output 1	Not available in CDF600-22xx		
Add.+1	D5	External output 2	Not available in CDF600-22xx		
Add.+1	D6	Result 1	Not available in CDF600-22xx		
Add.+1	D7	Result 2	Not available in CDF600-22xx		
Add.	D8	External Input 1	Input EXT. IN 1 on CDF600-22xx	X	X
Add.	D9	External input 2	Not available in CDF600-22xx		
Add.	D10	Sensor 1	Not available in CDF600-22xx		
Add.	D11	Sensor 2	Not available in CDF600-22xx		
Add.	D12	–	Reserved		
Add.	D13	–	Reserved		
Add.	D14	–	Reserved		
Add.	D15	Toggle **	Inverted after each read cycle. **	X	---

Observe byte order: D0 stands for the least significant bit of Add.+1

\* The bits are valid if the “Toggle” bit changes.

\*\* Not supported by every ID sensor.

The assignment of the Ctrl bits for the respective ID sensor is available in the respective GSDML documentation.



## 8.1.2 Ctrl bits Out

By adding the Ctrl bits Out, various functions can be activated in the ID sensor.  
For this to be allowed, the respective function must be configured in the ID sensor to “Fieldbus Input”.

Assignment of the Ctrl bits Out for the CLV62x ... 65x bar code scanner is used here as an example. The scope can vary depending on the ID sensor.

The Ctrl bits Out are only assigned in CDF600-22xx Proxy mode. In Gateway mode, the Ctrl bits Out are reserved.

Add.	Bit	Name	Meaning	Proxy mode	Gateway mode
Add.+1	D0	Trigger	Object trigger for the ID sensor	X	---
Add.+1	D1	Sensor Idle **	The attached ID sensor is put into sleep mode. **	X	---
Add.+1	D2	Teach In 1 **	The Teach In 1 teach-in process is triggered. **	X	---
Add.+1	D3	Teach In 2 **	The Teach In 2 teach-in process is triggered. **	X	---
Add.+1	D4	External Output 1	Not available in CDF600-22xx		---
Add.+1	D5	External output 2	Not available in CDF600-22xx		---
Add.+1	D6	Result 1	Not available in CDF600-22xx		---
Add.+1	D7	Result 2	Not available in CDF600-22xx		---
Add.	D8	–	Reserved		
Add.	D9	–	Reserved		
Add.	D10	–	Reserved		
Add.	D11	–	Reserved		
Add.	D12	Distance_Config_0 **	Controls bit 0 under dynamic (focus) switchover **	X	---
Add.	D13	Distance_Config_1 **	Controls bit 1 under dynamic (focus) switchover **	X	---
Add.	D14	Distance_Config_2 **	Controls bit 2 under dynamic (focus) switchover **	X	---
Add.	D15	Distance_Config_3 **	Controls bit 3 under dynamic (focus) switchover **	X	---

Observe byte order: D0 stands for the least significant bit of Add.+1

\*\* Not supported by every sensor.

The assignment of the Ctrl bits for the respective ID sensor is available in the respective GSD documentation.

*Example:*

Set CLV62x ... 65x to read cycle start via “Fieldbus Input” and read cycle stop via “Read Cycle Source”.  
The ID sensor can be triggered by the PLC using the bit D0 “Trigger”:

**Start/Stop of Object Trigger**

Start

Delay: 0 ms Fieldbus Input / CAN Open

Stop

Delay: 0 ms Trigger source or Not defined or Not defined

## 9 GSDML configuration in Proxy mode (optional)

In Proxy mode, the ID sensor can also be **firmly** configured using modules in the HW Config **as an option**.

The configuration modules of the ID sensor are added **after** the data channel / Ctrl bits. First, the module “**Start remote config**” must be selected and the module “**End remote Config**” must be added last. Depending on the configuration task, one or more configuration modules can be added.

Through this GSDML configuration, the scope of the configuration for a **typical task** can be handled with a **single ID sensor (stand-alone)**. Very extensive or special configurations must be carried out per SOPAS directly in the ID sensor.

### Important!

If the GSDML configuration is used, the **ID sensor is configured** every time the **PROFINET is restarted**. This cannot be suppressed in the ID sensor.

If the ID sensor is locally reconfigured using SOPAS in the meantime, the existing configuration is lost when PROFINET is restarted. In such a case, it may be necessary to change or remove the GSDML configuration in the PLC in advance.

Example: CLV61x bar code scanner

The screenshot displays the SIMATIC Manager HW Config interface for a SIMATIC 300 station. The hardware rack is configured with a CPU 315-2 PN/DP and various modules. The CLV61x bar code scanner is connected to the station. The 'PROFINET IO' tree on the right shows the configuration path for the CLV61x: CLV61x > 01\_Start Remote Config > 11\_Reading Config > 21\_2/5 Interleaved > 99\_End Remote Config. Three parameter windows are shown on the right, each with red arrows pointing to the corresponding configuration step in the tree. The first window shows 'General parameters' for the start of remote config. The second window shows 'General parameters' for the reading config. The third window shows 'General parameters' for the interleaved config. The bottom window shows the 'Allgemeine Parameter' for the end of remote config.

More information is available in the relevant GSDML documentation of the ID sensor.

## 10 Appendix

### 10.1 Quickstart ID sensor on the CDF600-22xx in Proxy mode via USB

The Quickstart describes the commissioning of a proxy-capable ID sensor with manual configuration via SOPAS:

- Connect the ID sensor to the 15-pin D-Sub-HD female connector of the CDF600-22xx.

SOPAS:

- In SOPAS, make sure that the required SDD file of the ID sensor has already been loaded.
- Set the CDF600-22xx to mode 0 and restart the CDF/ID sensor.  
The ID sensor can be configured via the USB interface of the CDF600-22xx.  
The Micro-B USB socket is located under the cover for the rotary coding switches.



Select the network wizard in SOPAS and connect USB. SOPAS shows the USB connection of the CDF600-22xx as the communication interface after scanning, **but the attached ID sensor itself is recognized as the device** and not the CDF600-22xx.

Network Scan Assistant		
Detected Devices	Communication Interface	Suitable Device Descriptions
CLV65x (station51)	USB - CLV65x - 12090051	CLV65x - V5.26

- Configure the ID sensor via SOPAS:
- Assign the planned PN name via SOPAS or later via HW Config on the sensor.
- Configure triggering. There are three options:
  - Trigger source = EXT. IN 1 → Triggering by photoelectric switch on EXT. IN 1
  - Trigger source = Fieldbus input → Triggering by trigger bit in Ctrl bits Out
  - Trigger source = (SOPAS) command → Triggering per function block
- Configure output format: Remove STX/ETX, not required.
- If necessary, adjust other parameters to suit the application, e.g., block code types that are not required, optimize read configuration.

PLC HW Config:

- Install the GSDML file of the ID sensor on the PLC. The sensors appear in the catalog under “PROFINET IO / Additional Field Devices / Ident Systems / SICK / ...”
- Add the required ID sensor in HW Config in PROFINET and assign the planned PN name. When adding the following modules are automatically supplemented:
  - Ctrl bits In
  - Ctrl bits Out
  - 32 byte input (can also be replaced by 8 – 128 byte input)
  - 32 byte output (can also be replaced by 8 – 128 byte output)
- If necessary, assign the planned PN name via HW Config on the sensor.

PLC program:

- Load function block and transfer I/O addresses. Note hex/dec representation.  
The function block is used to receive the read result data, regardless of the selected trigger option.  
The FB can also be used for triggering via command.
- Check the PROFINET communication (SF and BF LED on the CDF600-22xx and on the PLC)
- Check the triggering and data receipt

## 10.2 Quickstart ID sensor on the CDF600-22xx in Proxy mode without USB

The quickstart describes the commissioning of a proxy-capable ID sensor without using a USB interface.

- Connect the ID sensor to the 15-pin D-Sub-HD female connector of the CDF600-22xx (Mode 0). It is assumed that the ID sensor and the CDF600-22xx have the default settings.

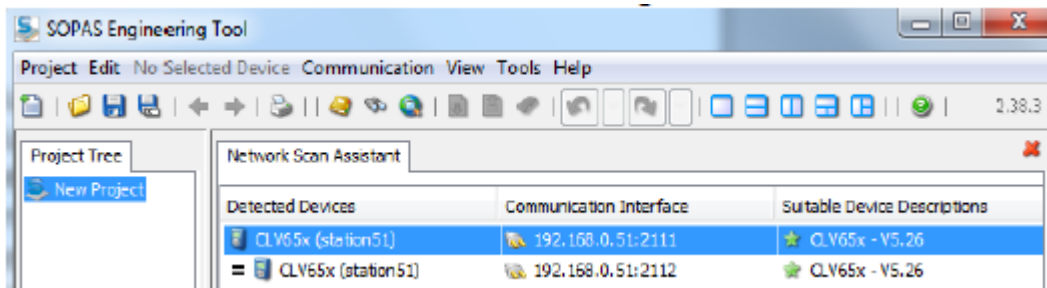
PLC HW Config:

- Install the GSDML file of the ID sensor on the PLC. The sensors appear in the catalog under “PROFINET IO / Additional Field Devices / Ident Systems / SICK / ...”
- Add the required ID sensor in HW Config in PROFINET and assign the planned PN name. When adding the following modules are automatically supplemented:  
Ctrl bits In  
Ctrl bits Out  
32 byte input (can also be replaced by 8 – 128 byte input)  
32 byte output (can also be replaced by 8 – 128 byte output)
- If required, configure the ID sensor via GSDML configuration by adding the required modules. See chapter 9.
- Scan for PN devices via HW Config and assign the planned PN name. See appendix, chapter 10.12.

If configuration via GSDML modules is not required, the ID sensor can also be configured using SOPAS through the network via TCP/IP:

SOPAS:

- In SOPAS, make sure that the required SDD file of the ID sensor has already been loaded. Reloading the SDD from the ID sensor may take some time. See Chapter 4.1
- Scan according to ID sensors via IP address: SOPAS shows the IP address of the CDF600-22xx as the communication interface after scanning, **but the attached ID sensor itself is recognized as the device** and not the CDF600-22xx.



- Configure the ID sensor via SOPAS:
- Configure triggering. There are three options:  
Trigger source = EXT. IN 1 → Triggering by photoelectric switch on EXT. IN 1  
Trigger source = Fieldbus input → Triggering by trigger bit in Ctrl bits Out  
Trigger source = (SOPAS) command → Triggering per function block
- Configure output format: Remove STX/ETX, not required.
- If necessary, adjust other parameters to suit the application, e.g., block code types that are not required, optimize read configuration.

PLC program:



- Load function block and transfer I/O addresses. Note hex/dec representation. The function block is used to receive the read result data, regardless of the selected trigger option. The FB can also be used for triggering via command.
- Check the PROFINET communication (SF and BF LED on the CDF600-22xx and on the PLC)
- Check the triggering and data receipt

## 10.3 Troubleshooting

This chapter provides some information on troubleshooting in the form of a condensed list.


### 10.3.1 Proxy mode troubleshooting

- Check the proxy capability of the attached ID sensor.  
Can this type/version be operated in Proxy mode on the CDF600-22xx?  
(See chapter 1.1) Check rotary coding switch on CDF600-22xx: Mode OK?
- Switch on the CDF600-22xx: “Power” LED must light up permanently → ID sensor has been detected.
- In SOPAS, read the system information of the ID sensor via USB → CDF600-22xx must have been recognized in the system status:

System Information							
Type	First time	Last time	Description	Info	State	Counter	Number
Information	8198:33:00	8198:33:00	Reset by Power On			1	0x200001E
Information	0:03:00	0:03:00	CDF600 detected	Firmware: V1.10.65		1	0x200070A

- Check the PN name and, if necessary, the IP address on the “PROFINET Proxy CDF600” tab under the device page “Network/Interfaces/IOs/Fieldbus Gateway”:

**Profinet Proxy CDF600**

Device Name   remanent ☒

Use 'PN Station Name' also for 'Device Name' ☒

PN state

PN IP-Address

PN Subnet-Mask

PN Default Gateway  remanent ☐

PN MAC-Address

- Check the PN name via HW Config or the PST tool. Name, IP address, and type OK?
- “Power” LED: Lights up continuously or flashes cyclically? Error codes, see Chapter 10.8.1  
Does the LED light up continuously when no PROFINET is attached?
- Check GSDML: Was the correct GSDML file of the ID sensor used?  
(not that of the CDF600-22xx) Modules correct or consistent with the communication mode?  
If necessary, remove configuration modules for testing purposes.
- PROFINET line OK? P1 Lnk and P2 Lnk LEDs OK?  
If necessary, check cable to another device.
- If necessary, use CDF600-22xx in Mode 2 and load the GSDML of the CDF600-22xx.  
Does the PROFINET work?

If no data arrives:

- Check modules in HW Config.
- Address for inputs and outputs correctly applied in the block? (Hex/dec representation)
- If no block is used, is the I/O address within the process image?  
Or get data with SFC 14/15.
- Heart bit appear?
- Does only the first data block appear and no others?  
Then the handshake does not work correctly? → Check addresses.  
Set handshake manually or with function block.

### 10.3.2 Gateway mode troubleshooting

- Check rotary switches on the CDF600-22xx: Mode correct?
- Switch on the CDF600-22xx: “Power” LED must illuminate → CDF600-22xx OK (Error codes, see Chapter 10.8.1)
- Check GSDML: Was the GSDML file of the CDF600-22xx used? (not that of the ID sensor)
- Check modules:
  - Ctrl bits In
  - Ctrl bits Out
  - 32 byte input (can also be replaced by 8 – 128 byte input)
  - 32 byte output (can also be replaced by 8 – 128 byte output)
- PROFINET line OK? P1 Lnk and P2 Lnk LEDs OK?  
If necessary, check cable to another device.
- Check the PN name via HW Config or the PST tool. Name, IP address, and type OK?
- Was the CDF600-2 reconfigured using SOPAS in regards to handshake and serial Cola A / Cola B protocol? The default setting is CDF600 with handshake and Cola A (standard STX/ETX frame).

If no data arrives:

- Check the attached device.  
Electrical connection OK on pin 2, 3, and 5 of the 15-pin D-Sub HD male connector?
- Data format of the attached device OK? (See chapter 7)
- Address for inputs and outputs correctly applied in the block? (Hex/dec representation)
- If no block is used, is the I/O address within the process image?  
Or get data with SFC 14/15.
- Heart bit appear?
- Does only the first data block appear and no others?  
Then the handshake does not work correctly? → Check addresses.  
Set handshake manually or with function block.

## 10.4 Overview of the mode switch of the CDF600-22xx

The operating mode rotary switch is located under the cover.

The CDF600-22xx must be restarted after each change.



Mode	Function
0	SICK devices (Proxy mode) 'DEVICE' RS-232 data transmission rate 57600 baud, parameter cloning active, GSD configuration possible
1	Reserved.
2	Other devices (Gateway mode) 'DEVICE' RS-232 data transmission rate 57600 baud
3	Reserved.
4	Other devices (Gateway mode) 'DEVICE' RS-232 data transmission rate 9600 baud
5	Reserved.
6	Reserved
7	Reserved
8	Reserved
9	Reserved
A	Reserved
B	Reserved
C	Reserved
D	Reserved
E	Firmware update for the CDF600-22xx via USB. PROFINET is not active
F	Transparent mode: Is used for a firmware update via USB or TCP/IP of the connected ID sensor 'DEVICE' RS-232 data transmission rate 57600 baud. PROFINET is not active



## 10.5 Read switch setting with GETDIAG via SOPAS terminal

Under certain circumstances (see below), the settings of the rotary coding switch and some additional details of the CDF600-22xx can be read using the GETDIAG command.

The command can be used via the **USB connection** as well as via a **TCP/IP connection** with the **IP address of the CDF600-22xx and port 2111 and 2112**.

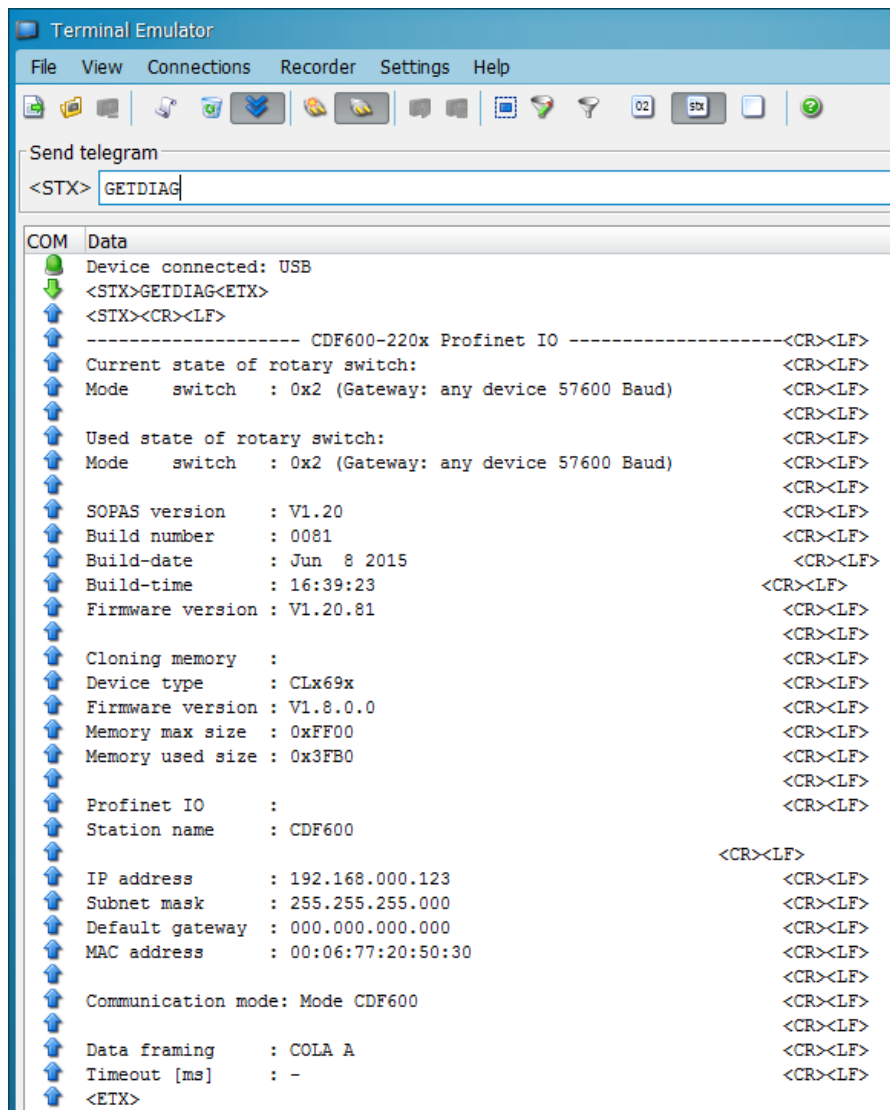
To do so, open the SOPAS terminal and establish a connection via USB or TCP/IP.

The "GETDIAG" command must be entered in the input line in uppercase.

As a response, the CDF600-22xx reports the current switch setting and the switch setting used when booting.

The command always displays the current switch status.

If the rotary coding switch is changed, the command must be sent again to display the current switch setting.



The screenshot shows a 'Terminal Emulator' window with a menu bar (File, View, Connections, Recorder, Settings, Help) and a toolbar. The 'Send telegram' field contains '<STX> GETDIAG'. The output area shows the following data:

```

COM Data
Device connected: USB
<STX>GETDIAG<ETX>
<STX><CR><LF>
----- CDF600-220x Profinet IO -----<CR><LF>
Current state of rotary switch: <CR><LF>
Mode switch : 0x2 (Gateway: any device 57600 Baud) <CR><LF>
Used state of rotary switch: <CR><LF>
Mode switch : 0x2 (Gateway: any device 57600 Baud) <CR><LF>
SOPAS version : V1.20 <CR><LF>
Build number : 0081 <CR><LF>
Build-date : Jun 8 2015 <CR><LF>
Build-time : 16:39:23 <CR><LF>
Firmware version : V1.20.81 <CR><LF>
Cloning memory : <CR><LF>
Device type : CLx69x <CR><LF>
Firmware version : V1.8.0.0 <CR><LF>
Memory max size : 0xFF00 <CR><LF>
Memory used size : 0x3FB0 <CR><LF>
Profinet IO : <CR><LF>
Station name : CDF600 <CR><LF>
IP address : 192.168.000.123 <CR><LF>
Subnet mask : 255.255.255.000 <CR><LF>
Default gateway : 000.000.000.000 <CR><LF>
MAC address : 00:06:77:20:50:30 <CR><LF>
Communication mode: Mode CDF600 <CR><LF>
Data framing : COLA A <CR><LF>
Timeout [ms] : - <CR><LF>
<ETX>
    
```

The content of the cloning memory which is used in Proxy mode to configure the ID sensor is also displayed. This configuration has no effect in Gateway mode.

The CDF600-22xx **cannot** respond to the GETDIAG command when:

- Mode E is selected. This mode is only used to update the firmware of the CDF600-22xx.
- Mode 0 is selected and an ID sensor is not yet attached or ready.
- The USB connection or TCP/IP connection is faulty.

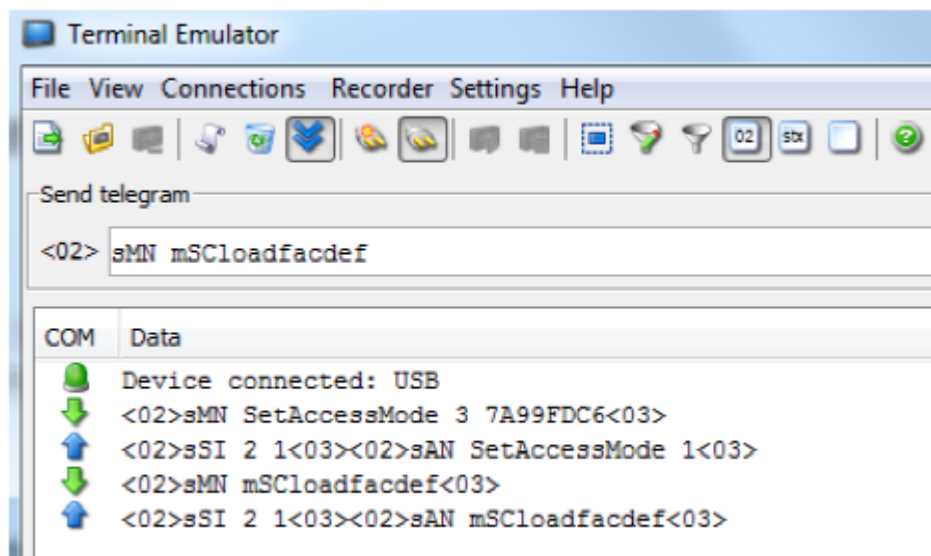


## 10.6 Resetting the CDF600-22xx to the default settings / clearing the cloning memory

In mode 2 (Gateway mode), the CDF600-22xx can be completely reset to the default settings by means of commands. This deletes all data such as PN name, IP address, and the cloning parameters for the ID sensor in Proxy mode.

Open the SOPAS terminal via USB connection and send the following commands. The CDF600-22xx must then be restarted:

```
sMN SetAccessMode 3 7A99FDC6
sMN mSCloadfacdef
```



The “GETDIAG” command can be used to check the result of the deletion process.

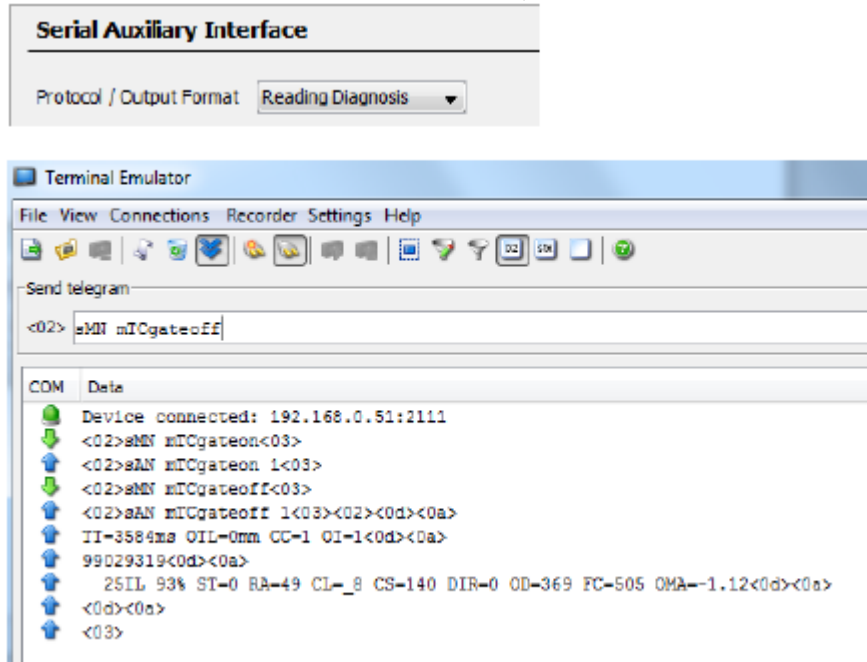
## 10.7 Monitoring data output via the SOPAS terminal

To do so, open the SOPAS terminal and establish a connection via USB or TCP/IP.

### Proxy mode:

In Proxy mode, the output of the serial AUX interface, e.g., read diagnostics, is displayed on the terminal, both via USB and TCP (port 2111/2112). Note that the output of the serial AUX interface is disconnected again by the ID sensor in Proxy mode each time it is started.

If SOPAS commands are entered in the terminal, these reach the ID sensor and not the CDF600-22xx.



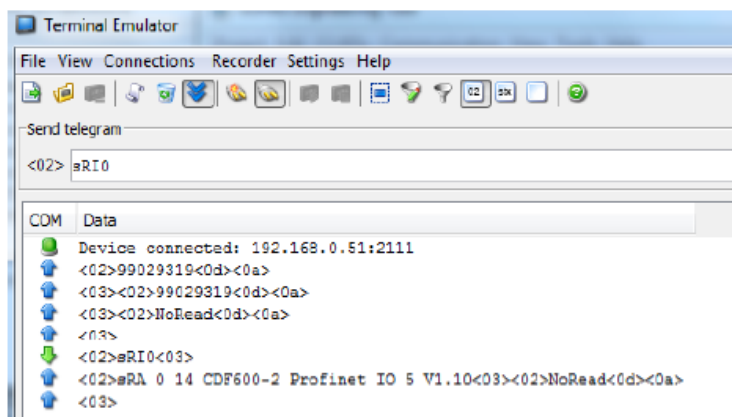
### Gateway mode:

Here, the terminal must be opened directly (Options / Terminal) via SOPAS ET instead of the connected CDF600-22xx.

In Gateway mode, the serial input data, e.g., output format #1 is displayed on the terminal, both via USB and TCP (port 2111/2112). Send data (commands) from the PLC are not shown.

If SOPAS commands are entered, only the CDF600-22xx is reached via **USB and on Port 2111** and not the ID sensor. If SOPAS commands are entered, the connected ID sensor is reached on **Port 2112**.

⇒ I.e., if the connected sensor can in general be serially configured using SOPAS. the sensor can also be reached and configured via the IP address of the CDF600-2 via Port 2112.



## 10.8 Function of the LEDs of the CDF600-22xx

Statuses and errors for the CDF600-22xx are indicated via 6 (M12 version) or 8 (AIDA version) status LEDs:

- POWER (green)
- EXT. IN 1 (yellow)
- SF (System Failure, red)
- BF (Bus Failure, red)
- P1 LNK/ACT (green/orange)
- P2 LNK/ACT (green/orange)



Once the CDF600-22xx is switched on, the EXT.IN1 LED lights up first and then all 6 LEDs light up together for 500 ms.

If mode E (firmware update of the CDF600-22xx) is active, all 6 LEDs flash together.

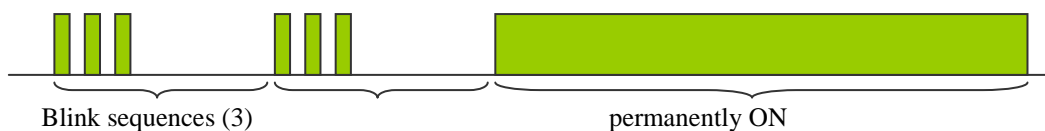
If mode F (firmware update of the connected ID sensor) is active, the Power LED flashes cyclically 3 times and the BF LED lights up.

LED status “ready for operation”:

The “Power” LED shows the status of the CDF600-22xx. If the LED is continuously ON and the “SF / BF” error LEDs are OFF, the CDF600-22xx is ready for operation and the PROFINET connection established.

### 10.8.1 Display of the Power LED

The flash sequence of the power LED indicates more diagnostic information.



POWER LED (green)	Meaning
ON	The CDF600-22xx is ready for operation (PROFINET communication does not have to be established yet) Proxy mode (mode 0): Communication with the attached ID sensor is established Gateway mode (mode 2 / 4): Communication with the ID sensor is not checked.
1	Proxy mode only (mode 0): CDF600-22xx is booting (device start) or searching for an attached ID sensor.
2	Position of the “Mode” rotary switch was changed during operation. This has no effect on operation, the CDF600-22xx only operates in the new operating mode once it has been restarted.
3	“Transparent operation” mode of the CDF600-22xx for updating the attached ID sensor. No PROFINET communication

### 10.8.2 Status of the “SF” LED

The “SF” LED indicates the internal status of the CDF600-22xx.

SF LED (red)	Meaning
OFF	CDF600-22xx without internal error
ON	Proxy mode (mode 0): CDF600-22xx is searching for an attached ID sensor.

### 10.8.3 Status of the “BF” LED

The “BF” LED indicates the status of the PROFINET.

BF LED (red)	Meaning
OFF	Data exchange OK, CDF600-22xx operates OK as PROFINET slave
ON	No connection to PROFINET controller (PLC). No data exchange. Possible reasons: <ul style="list-style-type: none"> <li>• Bus not connected</li> <li>• Master not available / OFF</li> <li>• Incorrect PROFINET name</li> <li>• Wrong GSDML file used</li> <li>• Wrong GSDML module selected</li> </ul>
Flashing (0.5 Hz)	<ul style="list-style-type: none"> <li>• Parameterization / configuration error in IO controller, no data exchange</li> <li>• Topology has been taught-in, device is connected so that it does not correspond to the taught-in topology. Ports P1 and P2 were reversed.</li> </ul>

### 10.8.4 Status of the LED “EXT. IN 1”

The LED “EXT. IN 1 ” is directly controlled by external input 1.

The LED also flashes twice briefly only when starting.

EXT.IN1 LED (yellow)	Meaning
OFF	External input 1 is not powered or open
ON	External input 1 is powered

### 10.8.5 “P1 LNK/ACT” status LED

P1 LNK/ACT LED	Meaning
OFF	The CDF600-22xx is not connected to any active network; no data traffic
ON (green)	Active network connected
ON (orange)	LED flickers when the CDF600-22xx is sending or receiving data, data traffic

### 10.8.6 “P2 LNK/ACT” status LED

P2 LNK/ACT LED	Meaning
OFF	The CDF600-22xx is not connected to any active network; no data traffic
ON (green)	Active network connected
ON (orange)	LED flickers when the CDF600-22xx is sending or receiving data, data traffic

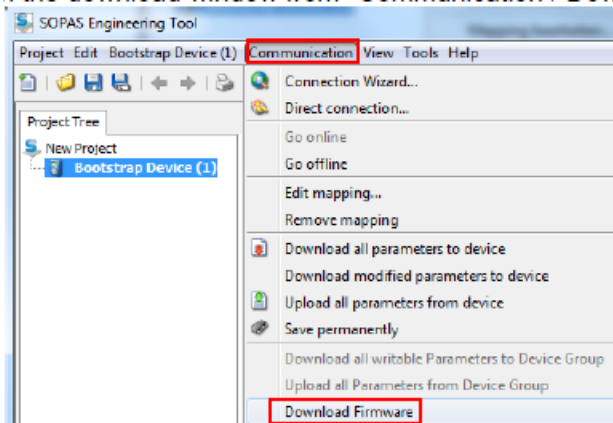
## 10.9 Firmware update of the CDF600-22xx (mode E) via SOPAS

In this mode, the CDF600-22xx can be updated in SOPAS via USB.

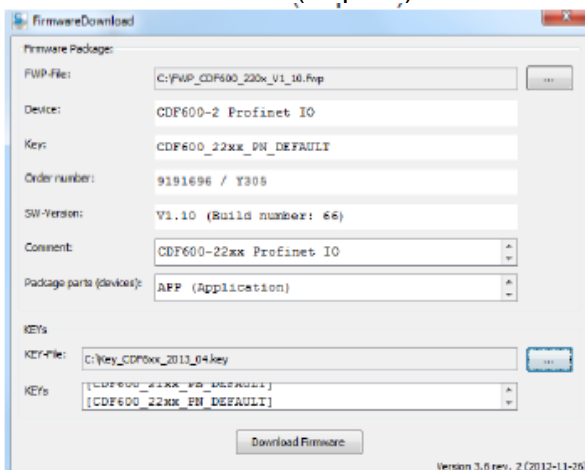
1. Set the “Mode” rotary coding switch to position “E” and switch the CDF600-22xx off and on.  
→ All the LEDs flash in this mode E.
2. Use the SOPAS Network Scan Assistant to scan the network → The CDF600-22xx is found via USB as a  
“Bootstrap Device”
3. Add attached CDF600-22xx to the project:



4. Open the download window from “Communication / Download Firmware”:

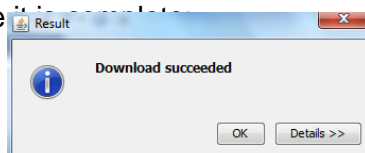


5. Select the current firmware (.fwp file) in the download window and the matching \*.key file:



6. Start download. Password for user-level service: “servicelevel”:

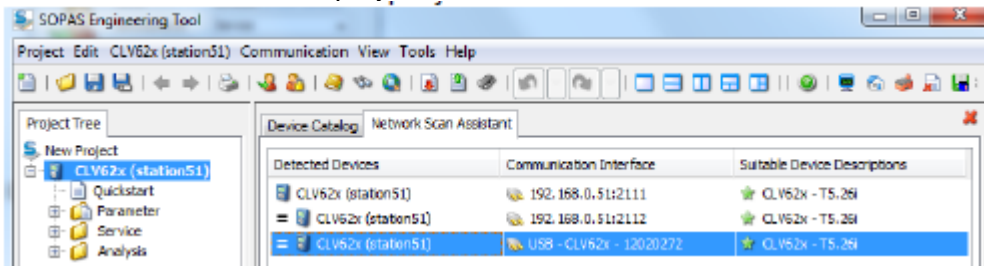
7. The download to the CDF600-22xx takes approx. 3 - 5 min. and the message “Download succeeded” will appear once



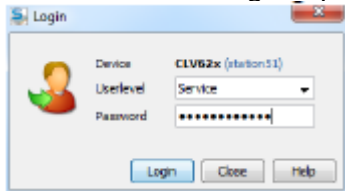
## 10.10 Firmware update of the attached ID sensor (mode F) via SOPAS

In this mode, an ID sensor attached to the CDF600-22xx (e.g., CLV61x, CLV62x-65x, LECTOR®62x, RFH6xx) can be updated in SOPAS via **USB** or via **TCP/IP (port 2111 and 2112)**. The CDF600-22xx uses the last used IP address. If necessary, this can also be changed using the SOPAS Auto IP Scan. The CDF600-22xx routes the USB and the TCP connections to the serial Aux port of the ID sensor.

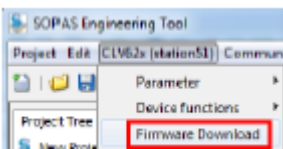
1. Attach ID sensor to the CDF600-22xx, set the “Mode” rotary coding switch to position “F”, and switch the CDF600-22xx off and on.
2. Use the SOPAS Network Scan Assistant to scan the network → The ID sensor is found via USB.
3. Add attached ID sensor to the project:



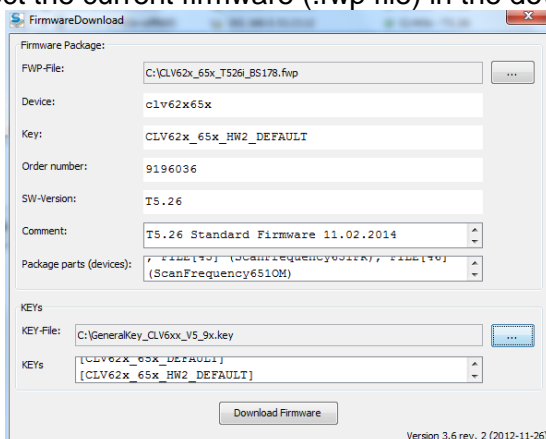
4. Log into the device using the password for user-level service: “servicelevel”:



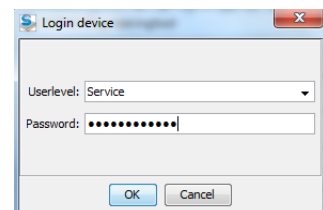
5. Use the command “CLV ... > Firmware Download” to open the download window:



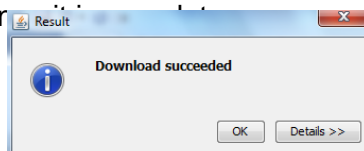
6. Select the current firmware (.fwp file) in the download window and the matching \*.key file:



7. Start download. Password for user-level service: “servicelevel”:



8. Depending on the ID sensor, the download can take up to approx. 40 min. and the message “Download succeeded” will appear on the screen:

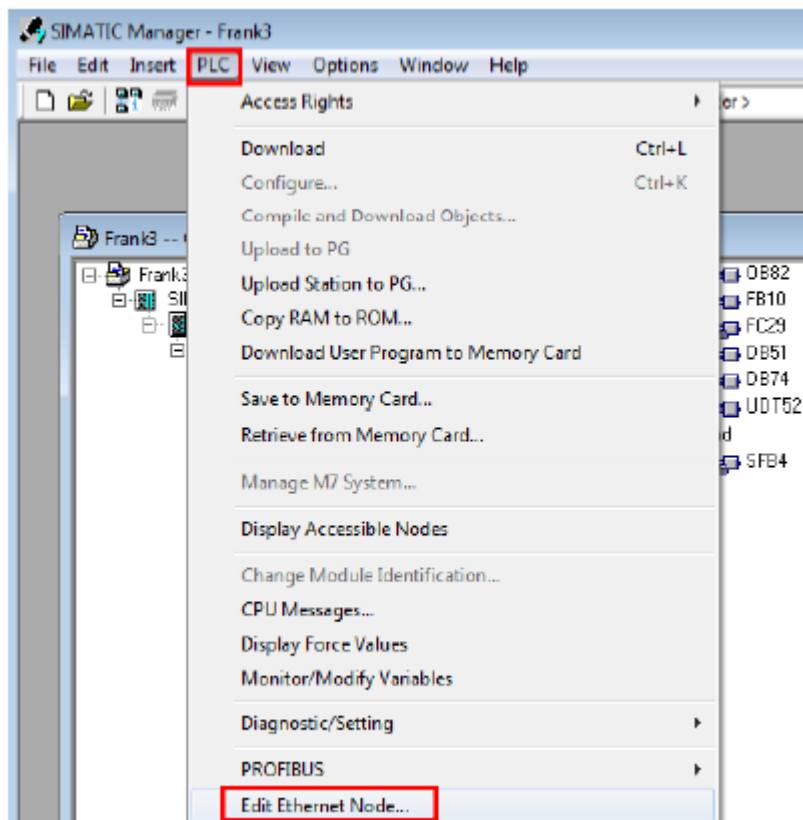


## 10.11 Software versions of the CDF600-22xx

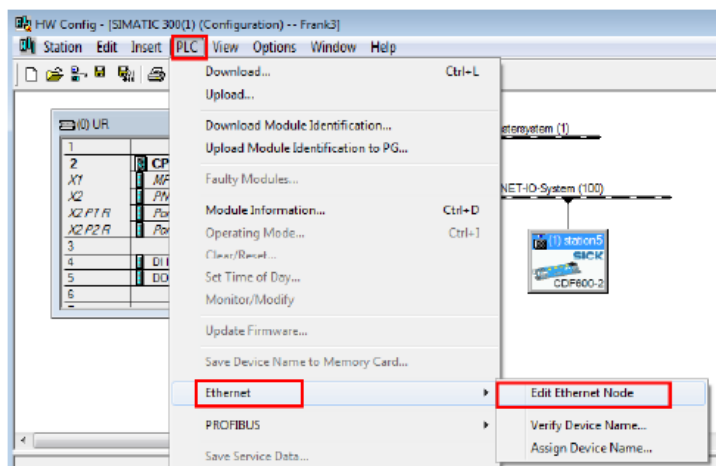
V1.00	First firmware version of the CDF600-22xx, → Update recommended
V1.04	Second firmware version of the CDF600-22xx, → Update recommended
V1.10	Third firmware version of the CDF600-22xx, contains all the functions that are listed in this documentation.

## 10.12 Tools for checking and assigning PN name and IP address from the PLC side

You can use this tool in the S7 manager to search for a PROFINET device and to check or change the PROFINET device name and IP address.



Or you can start it in the S7 HW Config:



This window opens in both cases:

You can scan the network using “Start”:

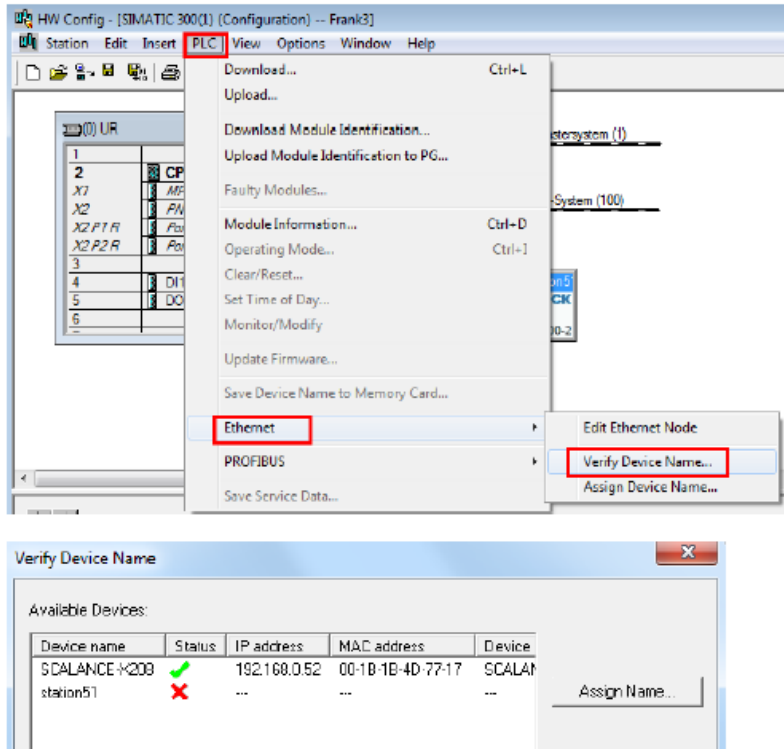
!	IP address	MAC address	Device type	Name	Subnet mask
	192.168.0.50	00-1B-1B-2D-7E-F7	S7-300	pn-io	----
	192.168.0.49	00-1B-1B-4D-77-17	SCALANCE X-200	scalance-x208	----
	192.168.0.51	00-06-77-06-0F-80	CDF600-220x	station51	----

You can search for a device (MAC address) and assign this a device name or an IP address.



## 10.13 Checking and, if necessary, assigning a PROFINET name from the PLC side via HW Config

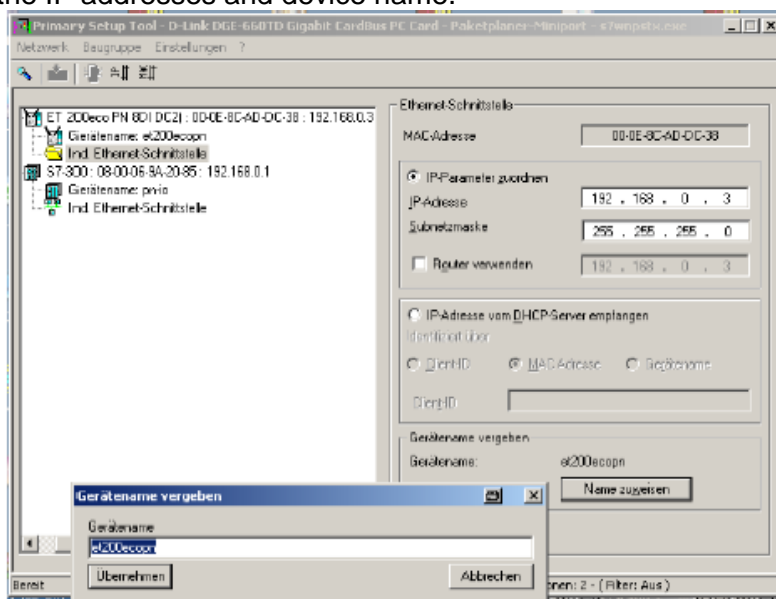
You can use this tool to check the PROFINET names of the PROFINET devices.  
Select the PROFINET line and start this function:



If some names are not available, a name can be assigned here.

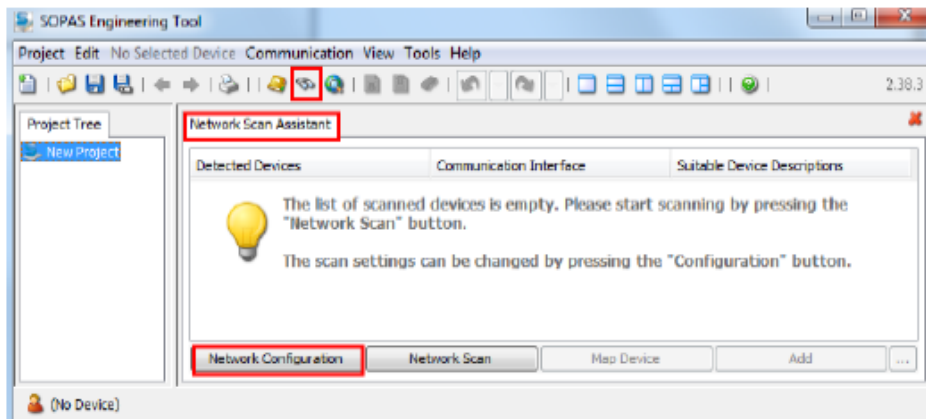
## 10.14 Checking the PROFINET name using the PST (Primary Setup Tool)

Alternatively, you can use the PST tool from Siemens to scan the PROFINET devices in the network and assign the IP addresses and device name.

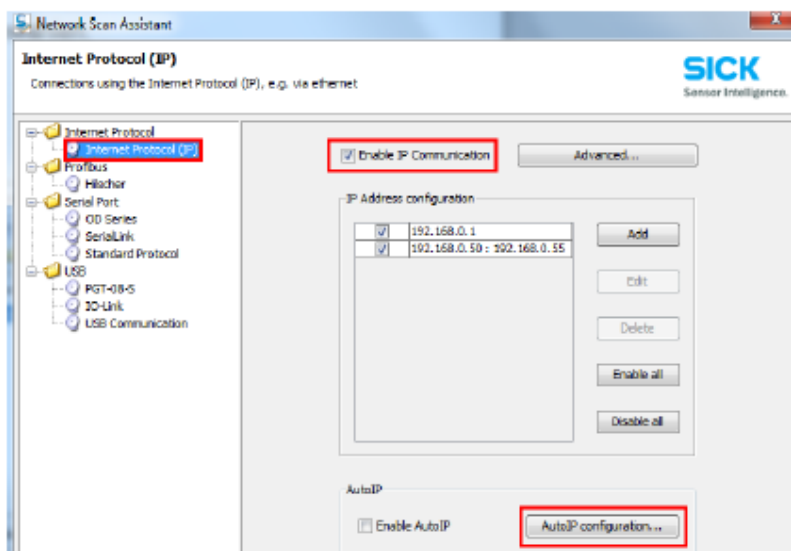


## 10.15 Searching the network using SOPAS Auto IP Scan

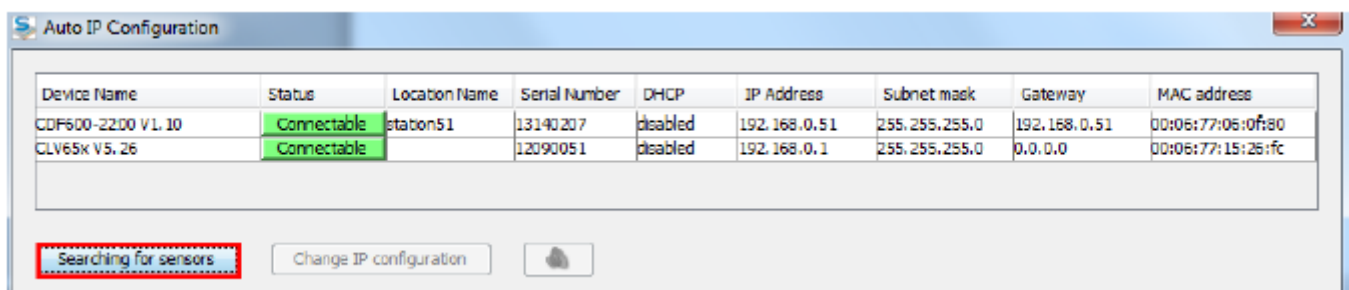
The SOPAS Auto IP Scan can also be used to scan the network for SICK devices. To do so, start a new project in SOPAS ET and call up the Network Scan Assistant:



Then enable IP communication and select the Auto IP settings:



Now you can search for SICK sensors. The search is based on MAC addresses and includes all SICK devices (MAC address = 00 06 77 xx xx xx), regardless of whether these are activated or not in PN. A CDF600-22xx is only found if the IP address is not 0.0.0.0.
















You can also change the IP address by double-clicking on the device. In the case of CDF600-22xx, the IP address can only be adjusted if the CDF600-22xx is no longer actively exchanging data with the PROFINET controller (PLC).

## 10.16 Configuration of a hand-held scanner from the IDM product family

To set a hand-held scanner of the IDM (IDM12x/x40/x41/x60/x61) product family to the required data format for Gateway mode 2, scan in the program codes below in order.

**Warning:** This deletes the previous configuration of the hand-held scanner.

Data format: STX data ETX RS232, 57600 baud, 8, n, 1

<b>Default</b> Warning: Resets all parameters to their default settings.	
<b>For hand-held scanners with Bluetooth only:</b> <b>Pair Mode</b>	
<b>For hand-held scanners with Bluetooth only:</b> Place the scanner on the smart cradle, then you will hear one short beep to indicate the pairing process is activated. The scanner will give continuous short clicks and the link indicator of scanner will flash blue quickly during the pairing process. When you hear 4 beeps in ascending tone, the pairing process is completed. You will see the link indicator of scanner giving <b>1 blue blink per 2.5 seconds</b> and the center indicator of the smart cradle turning <b>steady blue</b>	
<b>RS-232</b>	
<b>Program</b>	
<b>Baud rate</b>	
<b>Option code 8 (57600 baud)</b>	
<b>Data format</b>	
<b>Option code 0 (8,n,1)</b>	
<b>STX / ETX frame</b>	
<b>Option code 1</b>	
<b>Record suffix</b>	
<b>Option code 0</b>	
<b>FIN (Finish)</b>	

## 10.17 Notes regarding operation on other PROFINET controllers

If the CDF600-22xx is operated on a different controller to an S7, for which no ready-made function block is available, we recommend the following procedure:

- In Proxy mode, first set the mode to No-Handshake.
- First set triggering to hardware trigger or slow (e.g. 5 sec.) auto cycle for testing purposes and only examine the data receipt to start with.
- Use Hex representation for the first 10 input bytes: Can the Heartbeat bit be identified? Can the data be identified according to chapter 6.2.3., Example 5, receive telegram (NH mode)?  
Check the representation of bytes per integer/value variable. Byte order?
- Then, if necessary, switch to Handshake mode and a different trigger, e.g., trigger via fieldbus byte.

## 10.18 Changes Firmware version V1.20

Following are the important changes of version V1.20 compared to version V1.10:

- In version V1.10 in Gateway mode Port 2111 and 2112 were switched to CDF600-2.  
If a SOPAS terminal with the IP address of the CDF600-2 was opened both ports answered the CDF600-2.  
From version V1.20 and higher Port 2111 is still switched to the CDF600-2, but Port 2112 to the serial device. Thus a SOPAS capable sensor can be configured furthermore with SOPAS.
- The serial interface can be switched to ColaB (binary) in Gateway mode via SOPAS.  
During this the remote station must then also use this protocol, e.g.,: an MSC800 from V3.40 or higher.
- In Gateway mode it can also be switched to NoHandshake via SOPAS.  
The SDD file required for both CDF600-2 configurations (handshake and Cola B) can be loaded via SOPAS from mysick.com. An SDD upload from CDF600-2 is not possible.
- BF-LED now indicates by flashing if the topology is being taught-in, but the ports P1 and P2 have been reversed.
- Support from RFU6xx and CLV690 in proxy mode

(Blank page)

**Australia**

Phone +61 3 9457 0600  
1800 334 802 – tollfree  
E-Mail sales@sick.com.au

**Austria**

Phone +43 (0)22 36 62 28 8-0  
E-Mail office@sick.at

**Belgium/Luxembourg**

Phone +32 (0)2 466 55 66  
E-Mail info@sick.be

**Brazil**

Phone +55 11 3215-4900  
E-Mail marketing@sick.com.br

**Canada**

Phone +1 905 771 14 44  
E-Mail information@sick.com

**Czech Republic**

Phone +420 2 57 91 18 50  
E-Mail sick@sick.cz

**Chile**

Phone +56 2 2274 7430  
E-Mail info@schadler.com

**China**

Phone +86 4000 121 000  
E-Mail info.china@sick.net.cn

**Denmark**

Phone +45 45 82 64 00  
E-Mail sick@sick.dk

**Finland**

Phone +358-9-2515 800  
E-Mail sick@sick.fi

**France**

Phone +33 1 64 62 35 00  
E-Mail info@sick.fr

**Germany**

Phone +49 211 5301-301  
E-Mail info@sick.de

**Great Britain**

Phone +44 (0)1727 831121  
E-Mail info@sick.co.uk

**Hong Kong**

Phone +852 2153 6300  
E-Mail ghk@sick.com.hk

**Hungary**

Phone +36 1 371 2680  
E-Mail office@sick.hu

**India**

Phone +91-22-4033 8333  
E-Mail info@sick-india.com

**Israel**

Phone +972-4-6881000  
E-Mail info@sick-sensors.com

**Italy**

Phone +39 02 27 43 41  
E-Mail info@sick.it

**Japan**

Phone +81 (0)3 5309 2112  
E-Mail support@sick.jp

**Malaysia**

Phone +603 808070425  
E-Mail enquiry.my@sick.com

**Netherlands**

Phone +31 (0)30 229 25 44  
E-Mail info@sick.nl

**New Zealand**

Phone +64 9 415 0459  
0800 222 278 – tollfree  
E-Mail sales@sick.co.nz

**Norway**

Phone +47 67 81 50 00  
E-Mail sick@sick.no

**Poland**

Phone +48 22 837 40 50  
E-Mail info@sick.pl

**Romania**

Phone +40 356 171 120  
E-Mail office@sick.ro

**Russia**

Phone +7-495-775-05-30  
E-Mail info@sick.ru

**Singapore**

Phone +65 6744 3732  
E-Mail sales.gsg@sick.com

**Slovakia**

Phone +421 482 901201  
E-Mail mail@sick-sk.sk

**Slovenia**

Phone +386 (0)1-47 69 990  
E-Mail office@sick.si

**South Africa**

Phone +27 11 472 3733  
E-Mail info@sickautomation.co.za

**South Korea**

Phone +82 2 786 6321  
E-Mail info@sickkorea.net

**Spain**

Phone +34 93 480 31 00  
E-Mail info@sick.es

**Sweden**

Phone +46 10 110 10 00  
E-Mail info@sick.se

**Switzerland**

Phone +41 41 619 29 39  
E-Mail contact@sick.ch

**Taiwan**

Phone +886 2 2375-6288  
E-Mail sales@sick.com.tw

**Thailand**

Phone +66 2645 0009  
E-Mail tawiwat@sicksgp.com.sg

**Turkey**

Phone +90 (216) 528 50 00  
E-Mail info@sick.com.tr

**United Arab Emirates**

Phone +971 (0) 4 88 65 878  
E-Mail info@sick.ae

**USA/Mexico**

Phone +1(952) 941-6780  
1 (800) 325-7425 – tollfree  
E-Mail info@sick.com

**Vietnam**

Phone +84 8 62920204  
E-Mail Ngo.Duy.Linh@sicksgp.com.sg

More representatives and agencies  
at [www.sick.com](http://www.sick.com)