Technical Information

Subject: **Technical information of the CDF600-0300**

SICK Product: CDF600-0300

Department: Flexible Identification Creation Date: 2024/10/22

Version history

Version	Date	Author	Remarks
1	2024/10/22	Janis Meier	Initial Version

Table of content

1. Operating modes of the CDF600-0300

The CDF600-0300 has two different operation modes in which the CDF can be operated. The Proxy-Mode and the Gateway-Mode. However, it should be noted that the Proxy-Mode can only be used with certain sensors. While the Gateway-Mode can be used with all 4Dpro sensors with serial Aux.

1.1. Mode overview

The operation mode of the CDF600-0300 depends on the position of the “Mode” rotary coding switch. The “Mode” rotary coding switch is the first one shown in the illustration:

The following operating modes can be set:

No	Description
0	Sopas Proxy / CLV6xx (parameter cloning)
1	Sopas Proxy / CLV6xx (parameter cloning)
2	RS232 Gateway (57,6 kBd)
3	RS232 Gateway (57,6 kBd)
4	RS232 Gateway (9,6 kBd)
5	RS232 Gateway (9,6 kBd)
6	reserved
7	reserved
8	reserved
9	reserved
A	reserved
B	reserved
C	reserved
D	reserved
E	CDF FW up-date with 115kBd via AUX
F	transparent mode (57kBd, no CDF functionality, use for Sensor FW up-date)

1.2. Scope of functions in Proxy-Mode

Proxy-Mode (rotary coding switch “Mode” to 0) is not supported for all sensors. The ID sensors listed below are supported by the CDF in Proxy-Mode:

ID sensor	Model name	ESI file to be used
	CLV61x (V2.00 or higher)	SICK_CLV6xx.ESI.xml (can be found here: ESI file for CLV61x, CLV62x - CLV65x)
	CLV62x-65x (V5.10 or higher)	SICK_CLV6xx.ESI.xml (can be found here: ESI file for CLV61x, CLV62x - CLV65x)
	Lector620 (only V2.10)	LECTOR62x.ESI.xml (can be found here: ESI file for Lector620)

In this mode, the complete range of functions of the CDF600-0300 is available:

- The CDF is “integrated” in the connected ID sensor as “EtherCat Proxy CDF600” in SOPAS.
- The output format in SOPAS can be free defined. Output format 1 or 2 can be defined for the “EtherCat Proxy CDF600”.
- Handshake and No-Handshake mode can be used (when using Function block the Handshake mode is needed)
- The “EXT. IN1” switching input can be used directly to trigger the ID sensor, for example with a photoelectric sensor.
- In addition to the confirmed messaging data, the control bits for the PLC are available (Additional bits that can be used for the control and monitoring of the ID sensor, e.g. Good Read Bit or Trigger bit)
- 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 CDF600-0300. (Same as parameter CMC600 memory module)

- Configuration of the ID sensor with SOPAS via ethernet TCP/IP connection when the PC is directly connected to the network. In the CDF600-0300, this is implemented on the serial AUX interface of the ID sensor.

The following triggering types of the ID sensor are possible:

a) Triggering by photoelectric sensor or hardware signal at input “EXT. IN1” (configure ID-sensor to EXT. IN1)

b) Triggering by software trigger (configure ID sensor to command)

c) Triggering by trigger bit in the Ctrl bits (configure ID sensor to fieldbus input)

1.3. Scope 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 kBd (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 this mode the CDF operates as a protocol converter from RS-232 to EtherCAT. Only the fieldbus module is visible to the fieldbus master, the connected ID sensor is not recognized.

In Gateway-Mode, the range of functions of the CDF600-0300 is limited:

- The control bits are **not** available.
- Via **USB** and via **Ethernet Port 2111** only the CDF600-0300 itself can be configured with SOPAS.
  Port 2111 is connected logically with the CDF600-0300.
- 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.
- The output format must be set in Serial AUX settings in SOPAS. The output format must be STX/ETX framed. Output format 1 or 2 can be used.

In this mode, the Gateway-Mode ESI file must be used:

Sensor	ESI file to be used
Any sensor can be connected with serial data frame at pins 2 and 3 (AUX) with 57600 / 9600 Baud, 8 data bits, no parity and 1 stop bit	ESI file for Gateway-Mode

The following triggering types of the ID sensor are possible:

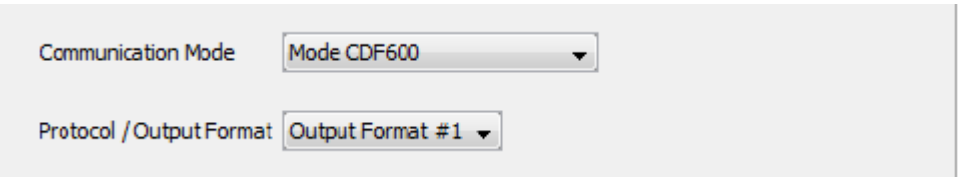
- Only triggering by software trigger command (configure ID sensor to command)

The PLC can detect the input “EXT. IN1” and “EXT. IN2” via bit D0 and D1 of the first input byte of the confirmed messaging, but the input can not be used for direct triggering of the ID sensor.

1.4. Overview of the communication protocols

In Proxy-Mode (mode 0):

The CDF600-0300 offers two communication protocols. The communication protocol can be set in the ID sensor under “Parameter / Network Interfaces IOs / Fieldbus Gateway”:



CDF600 Handshake mode / Confirmed Messaging (default, recommended)

- 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.
- Ctrl bits can be used to trigger or to set I/O's and to monitor.

CDF600 No Handshake Mode:

- 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

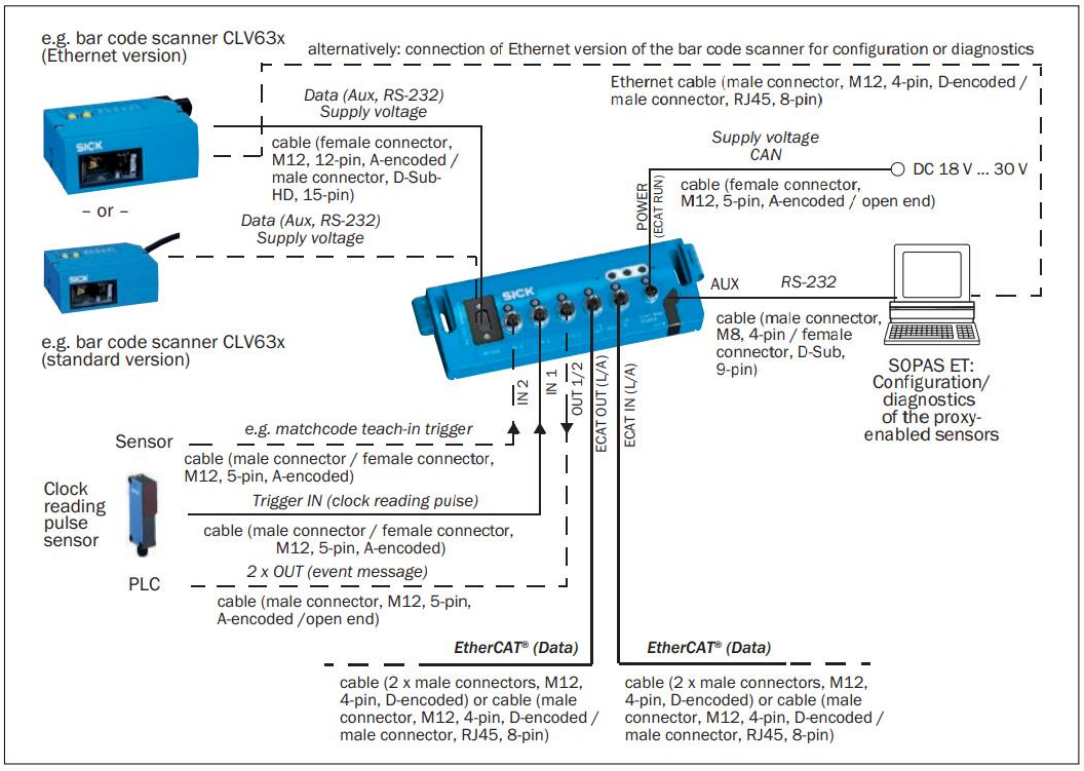
- Send and receive with max. telegram length of 4000 bytes (with blocking)
- A handshake is required on the PLC side.

CDF600 No Handshake Mode:

- 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.

2. Electrical connections and cables

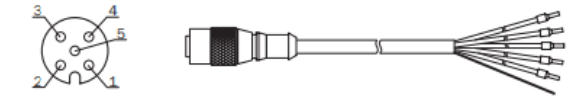
The figure shows the different connections of the CDF600-0300 with the required cables and sample connection with CLV63x.



In the following the pin assignments of the individual cables are shown.

Power connection:

Part number 6036384 (5 m):



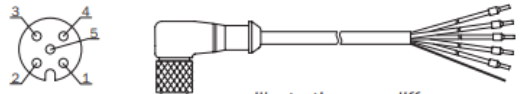
M12 - A / female connector

Illustration may differ

Pin	Signal	Function	Wire color
1	DC 24 V in	Supply voltage IN (DC 18 V ... 30 V)	Brown
2	CAN L	CAN bus [1)]	White
3	DC 0 V in	Supply voltage ground	Blue
4	CAN H	CAN bus [1)]	Black
5	N. c.	–	Gray
–	–	Shield	Metal

1) CAN-Bus only with support from SICK device with CAN interface

Part number 6049456 (3 m):



M12 - A / female connector

Illustration may differ

Pin	Signal	Function	Wire color
1	DC 24 V in	Supply voltage IN (DC 18 V ... 30 V)	Brown
2	CAN L	CAN bus 2)	White
3	DC 0 V in	Supply voltage ground	Blue
4	CAN H	CAN bus 2)	Black
5	N. c.	–	Gray
–	–	Shield	Metal

2) CAN-Bus only with support from SICK device with CAN interface

IN 1 and IN 2 connections:

Part number 6026133 (2 m) or 6026135 (10 m):

M12 - A / male connector

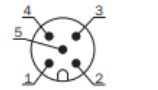
Illustration may differ

Pin	Signal	Function	Wire color
1	DC 24 V out	Supply voltage OUT (DC 18 V ... 30 V)	Brown
2	N. c.	–	White
3	GND	Ground	Blue
4	Ext IN 1 / Ext IN 2	External input 1 1) or external input 2 1)	Black
5	N. c.	–	Gray

1) Signal designation in the SOPAS ET configuration software of the SICK device that supports proxy mode. Function assignment and logics can be configured in the device.

OUT 1 / 2 connection:

Part number 6026133 (2 m) or 6026135 (10 m):



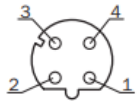
M12 - A / male connector

Illustration may differ

Pin	Signal	Function	Wire color
1	N. c.	–	Brown
2	Ext OUT 2	External output 2 [1)]	White
3	GND	Ground	Blue
4	Ext OUT 1	External output 1 [1)]	Black
5	N.c.	–	Gray

1) Signal designation in the SOPAS ET configuration software of the SICK device that supports proxy mode. Function assignment and logics can be configured in the device.

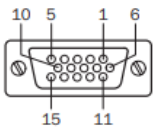
ECAT IN and ECAT OUT connections:



M12 - D / male

Pin	Signal	Function
1	TD+	Sender+
2	RD+	Receiver+
3	TD–	Sender–
4	RD–	Receiver–

Device connection:



Pin	Signal	Function
1	DC 24 V_{out}	Supply voltage OUT (DC 18 V ... 30 V), max. 0.8 A
2	TxD (AUX)	RS-232, sender
3	RxD (AUX)	RS-232, receiver
5	GND	Supply voltage ground
4, 6 ... 9	N. c.	–
10	CAN H	CAN bus [1]
11	CAN L	CAN bus [1]
12 ... 15	N. c.	–

1) CAN-Bus only with support from SICK device with CAN interface

AUX connection:

Part number 6021195 (2m) or 2027649 (8 m):

Pin	Signal	Function	Signal	Pin
4	TxD	AUX, RS-232	RxD	2
2	RxD	AUX, RS-232	TxD	3
3	GND	Ground	GND	5
1	N.c.	–	–	–

LEDs:

The LEDs above the M12 connectors show different events on the CDF600-0300.

LED - IN 1, IN 2:

Two digital inputs are integrated in CDF600-0300 HW (IN1, IN2). The inputs are opto-coupled with a common ground for these inputs. SICK light barriers can be connected directly.

LED state	Function
On	Signal output e.g. of the light barrier is High.
Off	Signal output e.g. of the light barrier is Low.

LED - OUT 1/2:

Two digital outputs are integrated in CDF600-0300 HW (OUT1/2). Two LEDs are use to signal high level on the outputs.

Out 1/2		Function
Green	**Orange**	
ON	OFF	OUT1 = **active**, OUT2 = inactive
ON	**ON**	OUT1 = **active**, OUT2 = **active**
OFF	OFF	OUT1 = inactive, OUT2 = inactive
OFF	**ON**	OUT1 = inactive, OUT2 = **active**

LED - ECAT OUT:

Describes the link/activity status of the physical EtherCAT OUT connection and activity on this connection.

LED state	Function
ON	Connection to EtherCAT network available, no activity on this connection, i.e. fieldbus module is offline.
OFF	Connection to EtherCAT network not available.
Flickering	Connection to EtherCAT network available, activity on this connection, i.e. fieldbus module is online.

LED - ECAT IN:

Describes the link/activity status of the physical EtherCAT IN connection and activity on this connection.

LED state	Function
ON	Connection to EtherCAT network available, no activity on this connection, i.e. fieldbus module is offline.
OFF	Connection to EtherCAT network not available.
Flickering	Connection to EtherCAT network available, activity on this connection, i.e. fieldbus module is online.

LED - ECAT RUN:

Describes the EtherCAT state in which the CDF600-0300 is currently operating.

LED state	Slave State	Function
OFF	Initialization	The device is in state INIT.
Blinking	Pre-Operational	The device is in state pre-Operational.
Single Flash	Safe-Operational	The device is in state Safe-Operational.
ON	Operational	The device is in State Operational.
Flickering	Initialization or Bootstrap	The device is booting and has not yet entered the INIT state or The device is in state Bootstrap. Firmware download operation in progress.

3. ID sensor connection

3.1. Connecting Lector62x, CLV61x, CLV62x – 65x, RFH6xx and RFU6xx

The ID sensors: Lector621, RFH6xx and RFU6xx can only operate in **Gateway-Mode**. This means that the mode rotary switch of the CDF must be set to 2. The ID sensors: Lector620, CLV61x and CLV62x - 65x can be operated in **Proxy-Mode** (rotary switch set to 0) or in **Gateway-Mode** (rotary switch set to 2). The ID sensors can normally be connected to the CDF via the standard Device cable.

3.2. Connecting Lector611

The Lector611 can only be operated in **Gateway-Mode** (rotary switch set to 2). With the Lector611, the exception is that the Lector611 does not have serial AUX. Since the serial data transmission via the device cable runs via serial AUX, there are two alternative options for the Lector611:

1. Special device cable:

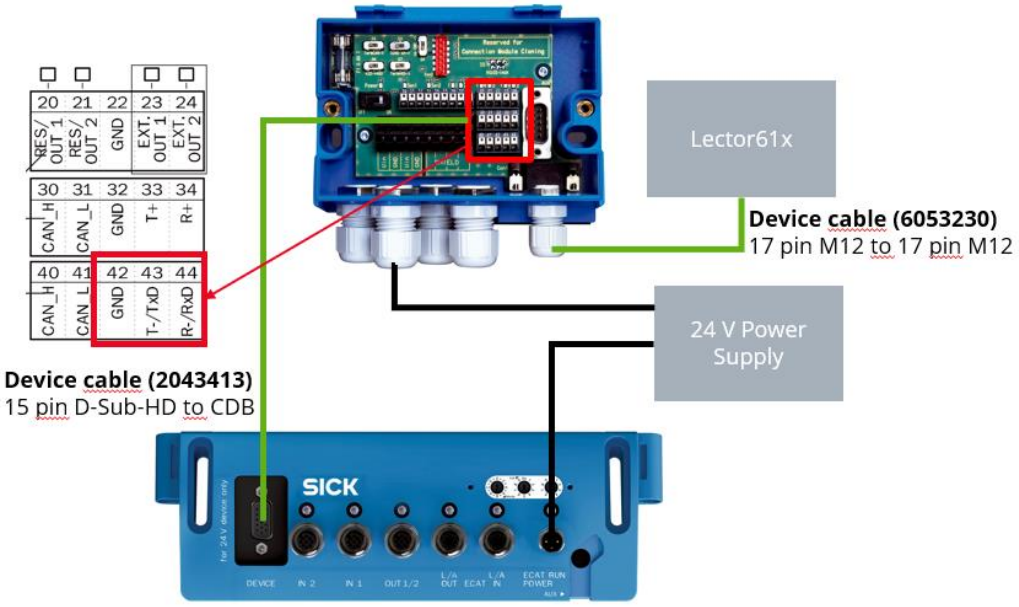
A special device cable for the Lector611 can be used instead of the standard device cable. This cable has serial AUX and Host swapped so that the data can be transferred via the Host interface. Part number: 2127037.

2. Connect with CDB650

The Lector611 can be connected to a CDB650. The serial Host from the Lector611 can then be routed to the CDF600-0300. To do this, the cable from the CDF to the CDB650 must be connected as follows:

CDF600-0300 to CDB:

TxD, pin 2 -(white)----------> ---- RxD, terminal 44
RxD, pin 3 -(brown)--<---------- TxD, terminal 43
Gnd, pin 5 -(blue)---------- Gnd, terminal 42

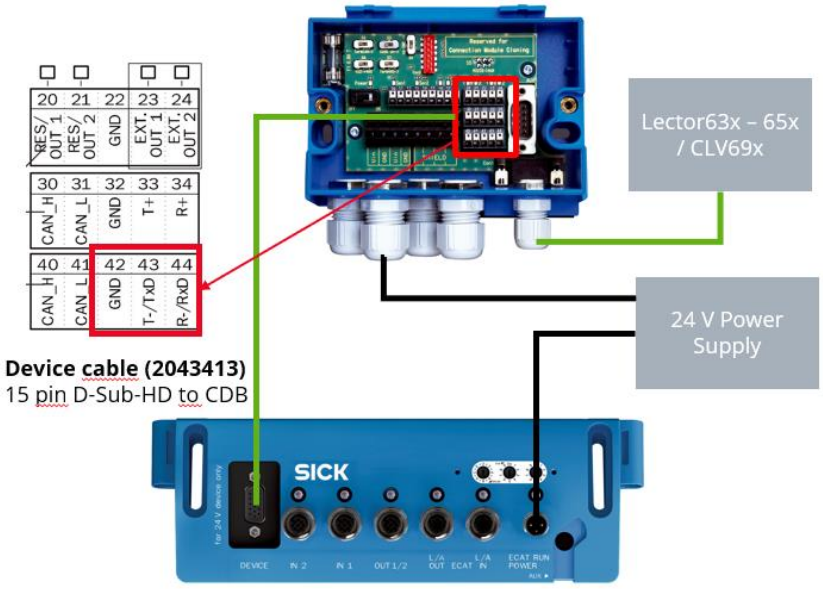


3.3. Connecting Lector63x – 65x & CLV69x

The problem with the Lector63x - 65x and CLV69x devices is that the CDF600-0300 can only supply a maximum current of up to 0.8A to the devices. Unfortunately, this is not enough for the Lector63x - 65x and CLV69x. For this reason, the ID scanners must be connected to a CDB650 or CDM420 with a 2 A fuse. The CDB650 and the CDM420 can supply enough current to operate the Lector63x - 65x and CLV69x. Therefore, the serial Host must be wired from the CDB/CDM to the CDF (15-pin D-Sub to open cable end).

CDF600-0300 to CDB/CDM:

TxD, pin 2 -(white)----------> ---- RxD, terminal 44
RxD, pin 3 -(brown)--<---------- TxD, terminal 43
Gnd, pin 5 -(blue)---------- Gnd, terminal 42



3.4. Connecting a handheld 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.

Set the CDF600-0300 to mode 2 (Gateway-Mode).

The CDF600-0300 is then fixed to 57600 baud and the communication protocol is set to CDF600.

3.5. Connecting other devices with RS-232 interface

If the CDF600-0300 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 CDF600-0300 always adds the STX/ETX frame.
This cannot be suppressed.

CDF600-0300:

15-pin HD socket device: External RS-232 device:

TxD, pin 2 ----------->----- RxD

RxD, pin 3 ----<---------- TxD

Gnd, pin 5 ---------- Gnd

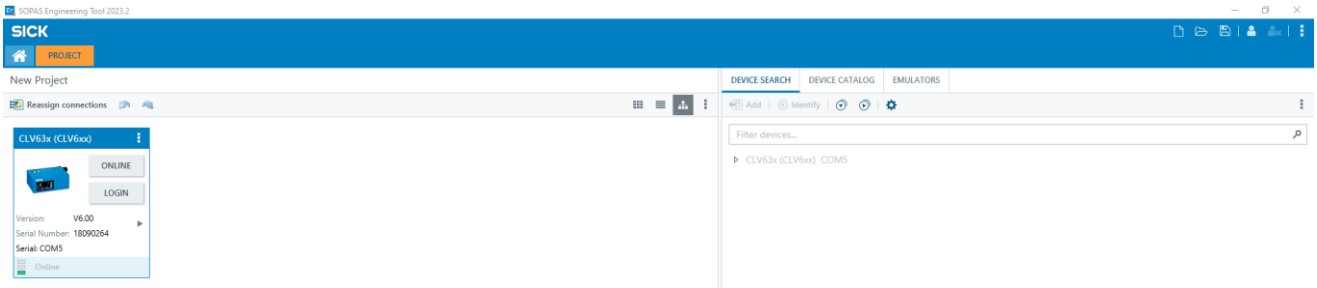
In Gateway-Mode, the USB interface of the CDF600-0300 cannot be used to configure the connected ID sensor. The ID sensor must then be configured, for example, via a direct Ethernet interface or USB.

4. Configuration via USB on the CDF600-0300

4.1. Configuring the ID sensor via USB on the CDF600-0300 in Proxy-Mode

In Proxy-Mode (rotary switch in position 0), it is possible to parameterize the connected ID sensor via USB connection. This requires a cable that converts RS-232 to USB (cable article number: 6034574). This cable can be connected from the AUX port of the CDF (see image below) to a PC.

To be able to find the ID sensor via SOPAS ET, the correct search setting must be set in SOPAS ET. The search settings must be set to Serial communication (57.6 kBaud). If the search settings are correct, the ID sensor itself will be found in SOPAS ET and not the CDF.



Once the connection has been established, the device window of the ID sensor can be opened by double-clicking.

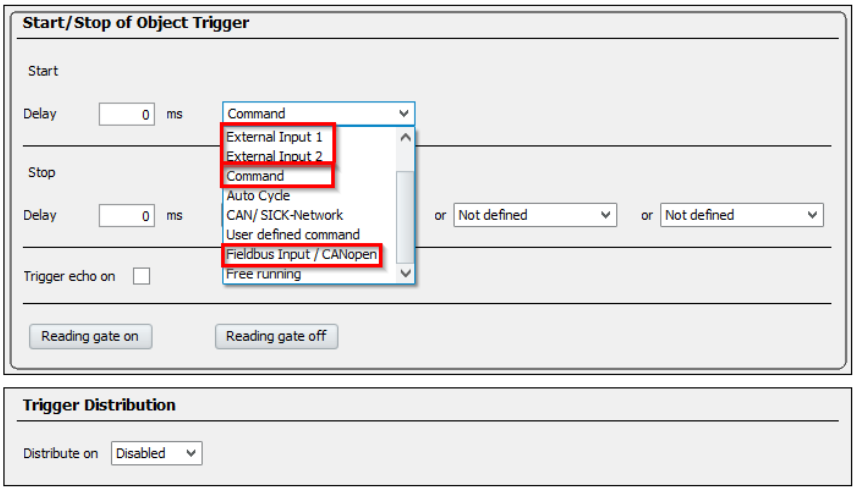
For the CDF600-0300, no direct configuration is required, **i.e., the connected ID sensor automatically recognizes the CDF600-0300.** SOPAS shows this as a system status in the "System Information" display window of the ID sensor as "CDF600 detected". The firmware version of the CDF600-0300 is also displayed.

Information	0:03:00	0:03:00	CDF600 detected	Firmware: V1.20.32		1	0x200070A

Trigger:

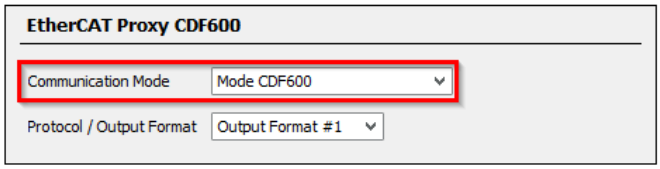
As the CDF is used in Proxy-Mode, the following trigger settings are possible:

- External input (hardware trigger)
- Command (software trigger)
- Fieldbus Input (Ctrl bits)



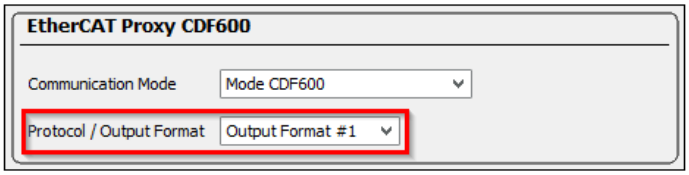
Communication Mode:

The communication mode of the CDF can be changed under “EtherCAT Proxy CDF600”. It is possible to use the CDF in handshake mode (Mode CDF600) or in no handshake mode (Mode CDF600 no handshake). If the communication mode is changed from the default, the parameters must be saved permanently, and the CDF must be restarted.



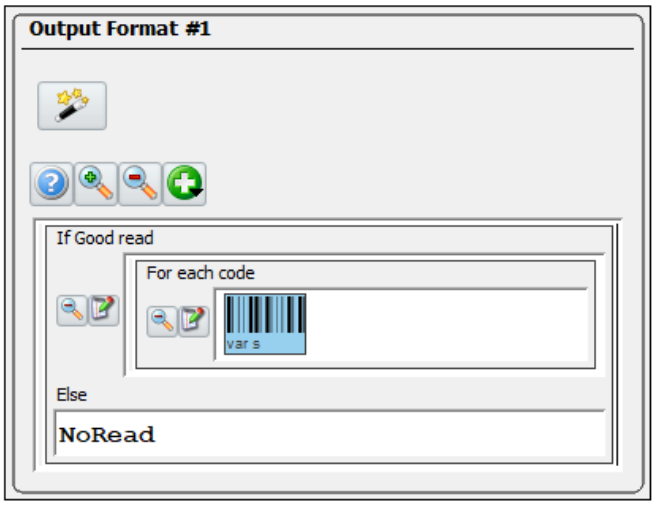
Output format:

In addition to the communication mode, the output format to be used must also be defined. It can be decided between Output Format #1 and Output Format #2.



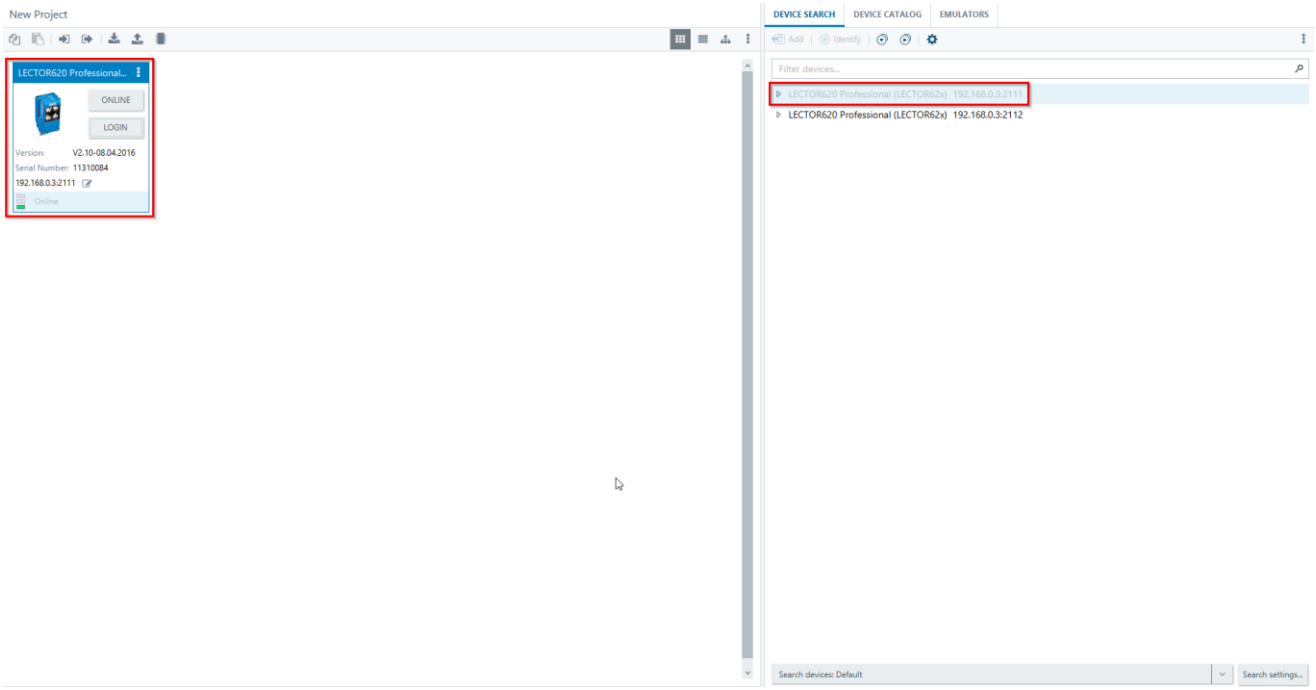
The required output format must also be set. The format does not require any STX/ETX frame because transport works in Proxy-Mode without this framing. If STX/ETX is set as the frame, the characters are entered in the data for the PLC as control characters, which is not normally desirable.

Set output format **without** STX/ETX in Proxy-Mode:

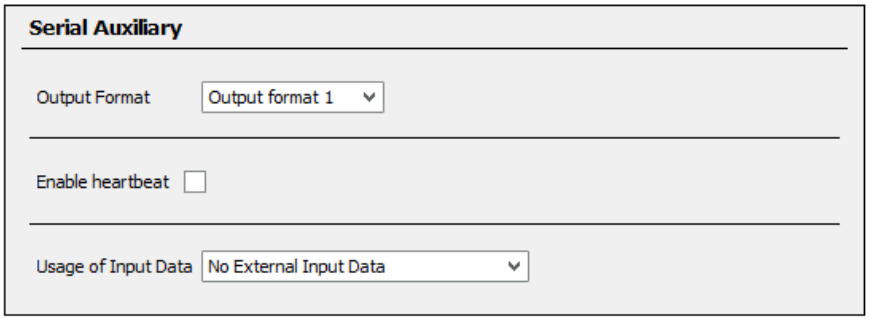


4.2. Configuring the CDF600-0300 in Gateway-Mode

For Gateway-Mode, the rotary switch of the CDF600-0300 must be set to 2 (57.6 kBd) or 4 (9.6 kBd). To be able to parameterize the ID scanner, the ID scanner must be configured via its Ethernet interface, for example.

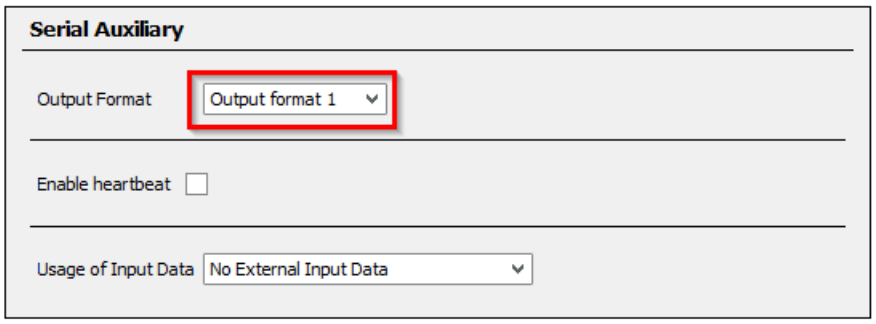


The connected ID scanner communicates with the CDF via the serial AUX interface (With the exception of the Lector611, which only has a serial host interface). The settings of the serial AUX interface must therefore be adjusted accordingly in the ID scanner.

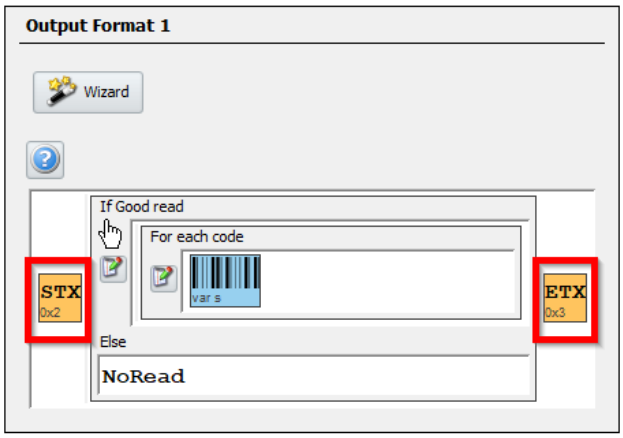


Output format:

The output format can also be selected via the serial AUX window.

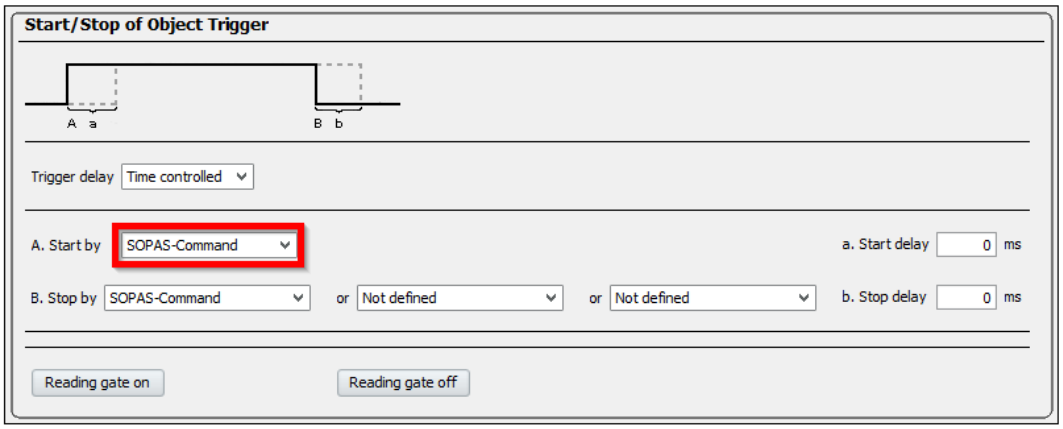


Please note that the output format STX/ETX must be framed.



Trigger:

Only the software trigger is possible in Gateway-Mode. Therefore, the trigger settings in the Object Trigger Control must be set to Command.



4.3. Parameter cloning in CDF600-0300 in Proxy-Mode

The CDF600-0300 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 CDF600-0300. If it is, then this is also stored and used on the ID sensor. If the parameter set on the CDF600-0300 is empty when it is switched on (delivery state), then the ID sensor stores its current parameters on the CDF600-0300. When it is next switched on, the ID sensor loads this again from the CDF600-0300.

During the configuration of the ID sensor, the parameters are stored on the ID sensor and **also on the CDF600-0300 every time they are stored permanently**. As a result the storage process takes longer. In this way the parameter set on the CDF600-0300 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 are transferred directly from the CDF600-0300 to the ID sensor. The previous configuration of the exchange unit (ID sensor) is not relevant and is overwritten.

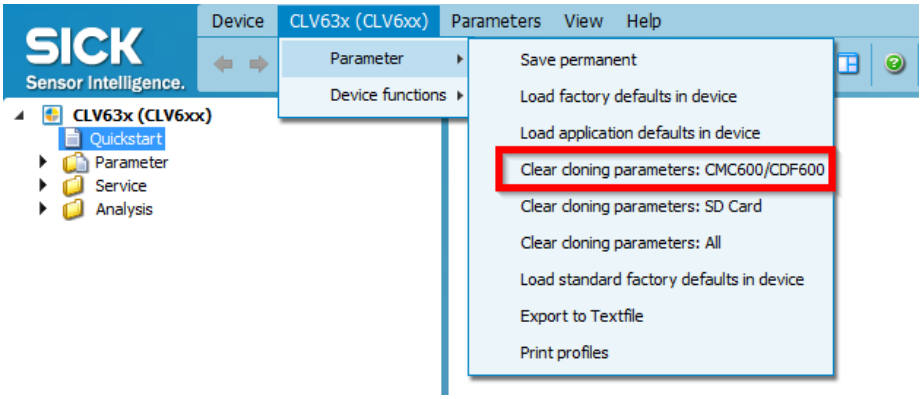
CDF600-0300 replacement:

If the CDF600-0300 is replaced with a device of the same type, it is important that the new CDF600-0300: does not contain any cloning parameters. If necessary, this can be checked by using the command “sRN CdfParaDevType” via Terminal in mode 2. See appendix, chapter 9.2.

If the parameter set on the CDF600-0300 is empty when it is switched on (delivery state), then the ID sensor stores its current parameters on the CDF600-0300. When it is next switched on, the ID sensor loads this again from the CDF600-0300.

Clearing the cloning memory in the CDF600-0300 via SOPAS:

In Proxy-Mode, the cloning memory in the CDF600-0300 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 CDF600-0300 can be cleared in mode 2 via a command. See appendix, chapter 9.2.

5. Handling on PLC side

This chapter shows how a CDF is integrated into a PLC in Gateway- or Proxy-Mode. The examples in this chapter relate to integration into a Beckhoff PLC with TwinCAT3.

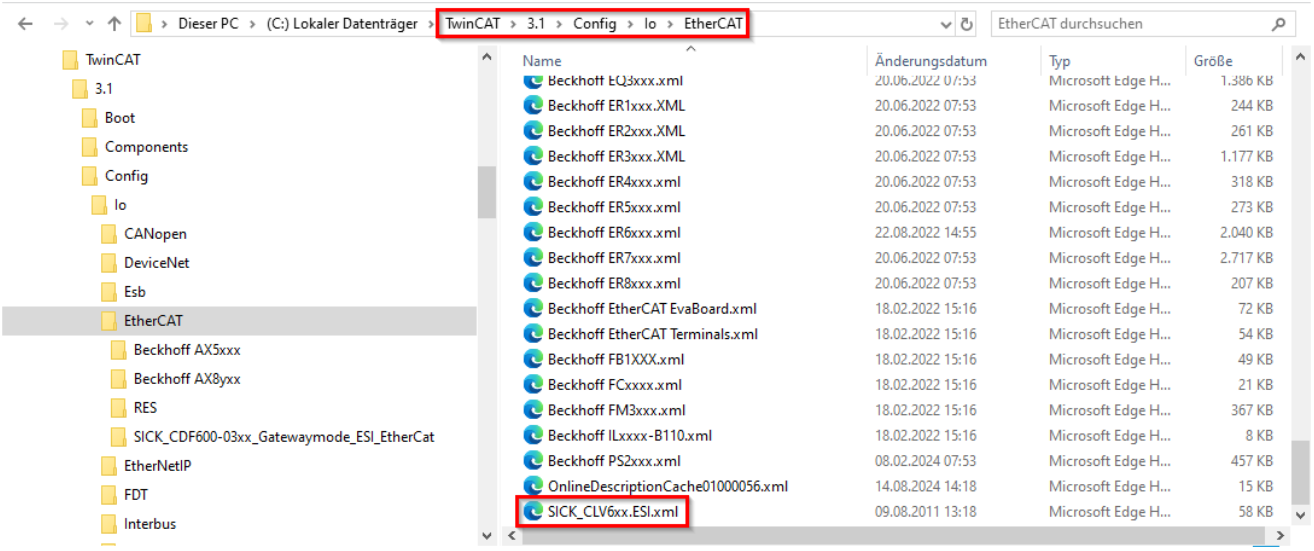
5.1. Integration of the ESI file

First of all, the appropriate ESI file must be downloaded, see chapters 1.2 and 1.3 for links. Depending on the type of ID sensor and the operating mode of the CDF, a different ESI file is required.

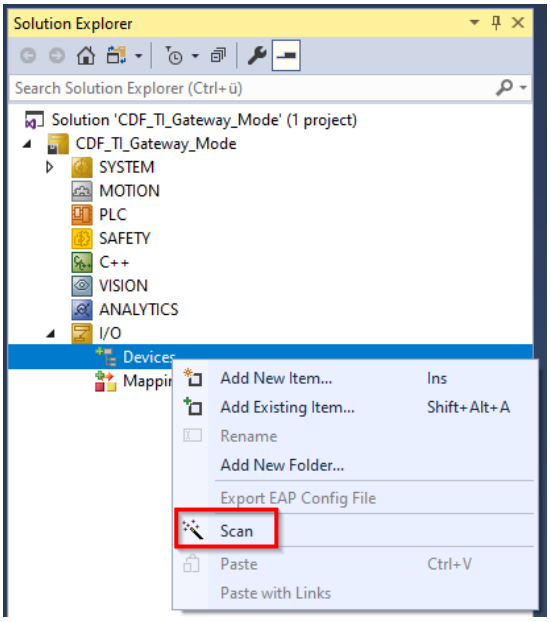
As soon as the appropriate ESI file has been downloaded, it must be integrated into the TwinCAT3 environment. To do this, the ESI file must be stored in the following path:

TwinCAT\3.1\Config\Io\EtherCAT

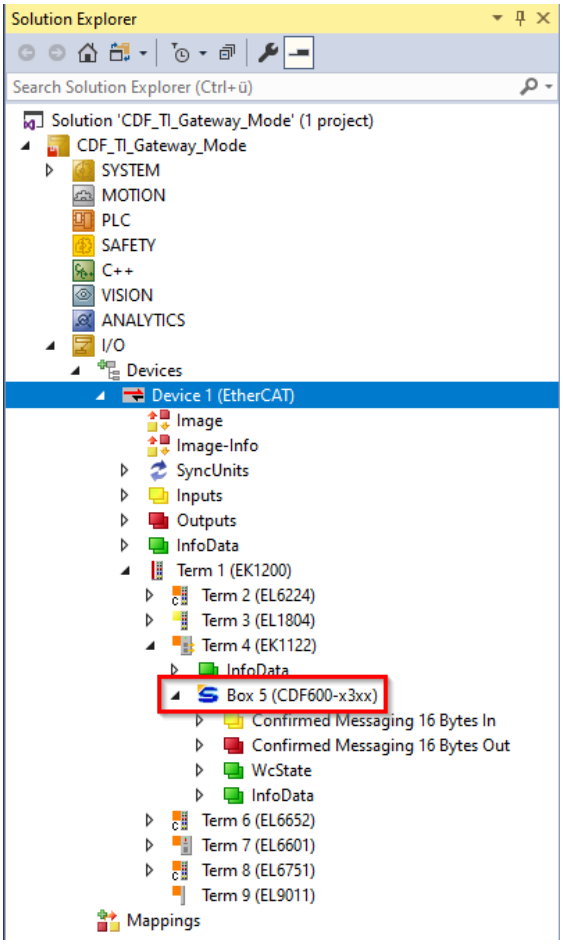
The figure shows an example of storing the ESI file of the CLV Proxy-Mode.



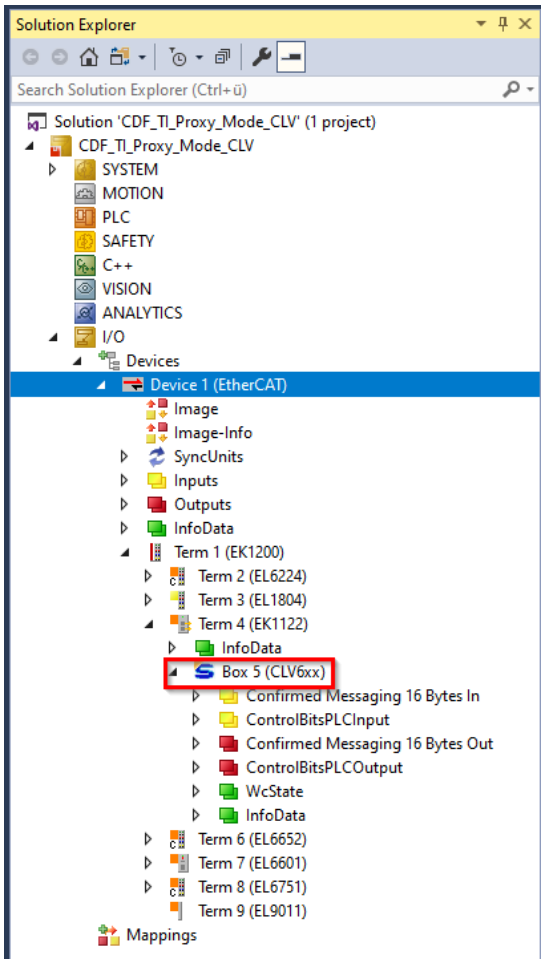
TwinCAT3 can then be started. A device scan can then be carried out via I/O → Devices → Scan and the CDF is recognized at the connected terminal:



Example CDF in Gateway-Mode:

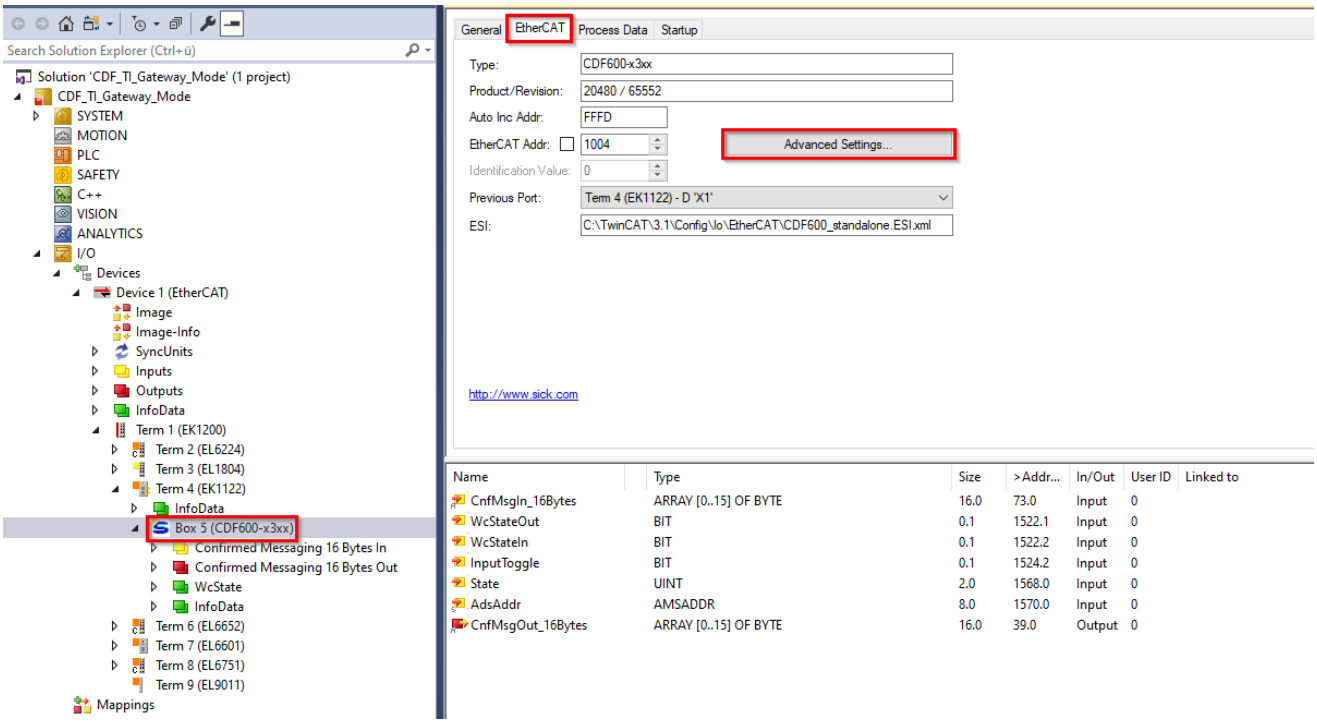


Example CDF in Proxy-Mode (CLV):

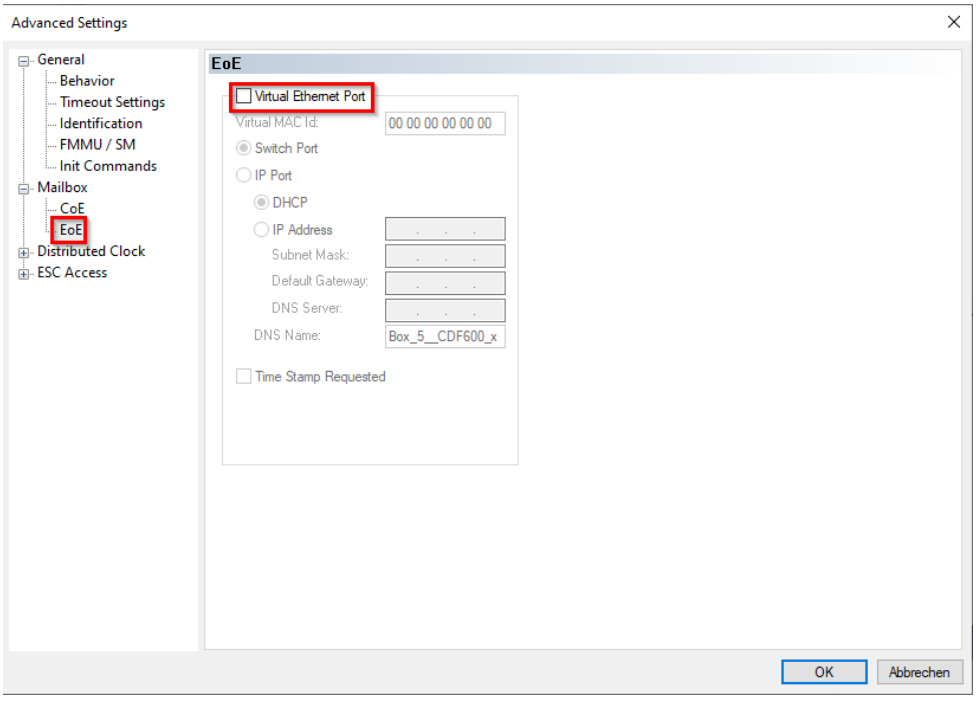


5.2. EoE settings

EoE is used for parameterization of the sensor via PLC to the CDF. EoE is not used for the operational data exchange. EoE must therefore be deactivated. To deactivate EoE, the advanced settings must be opened on the CDF under EtherCAT.



Navigate to Mailbox → EoE to remove the tick there.

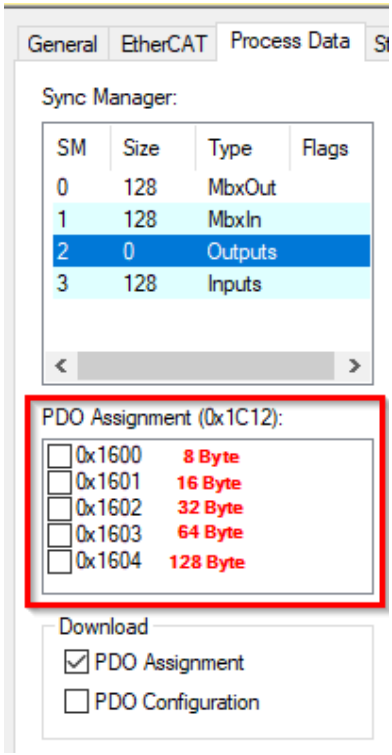


5.3. PDO configuration

The size of the Confirmed Messaging input and output data can be customized by the user. It is possible to choose between 8, 16, 32, 64 and 128 bytes. It should be noted that 5 bytes always belong to the header.

Example: 16 bytes = 5 bytes header + 11 bytes data

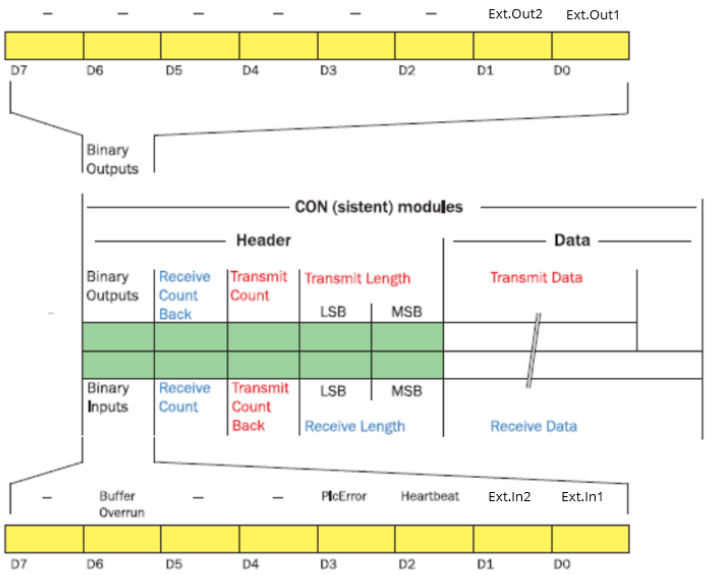
The user can select the different sizes of the input and output data via the PDO assignment:



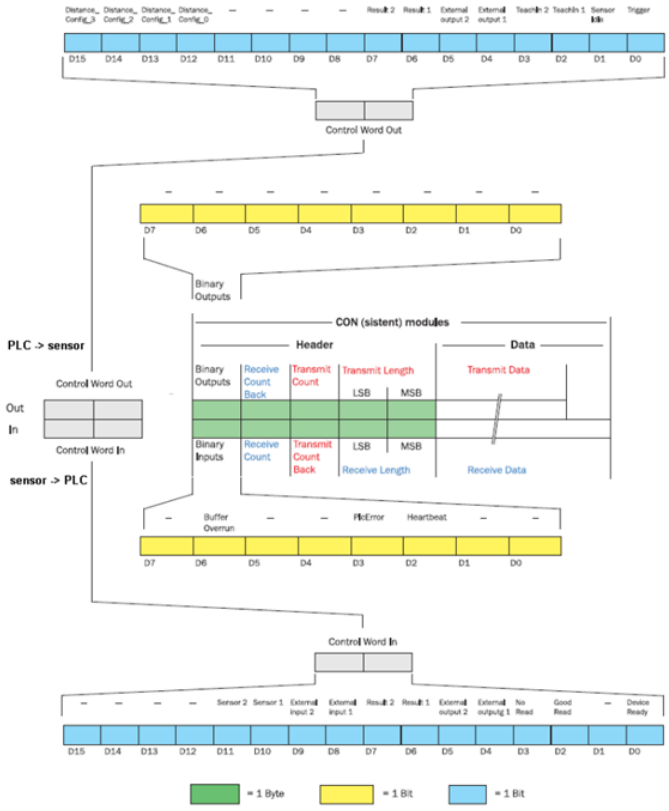
5.4. Communication from sensor to PLC

Data transmission from the sensor to the PLC is as follows:

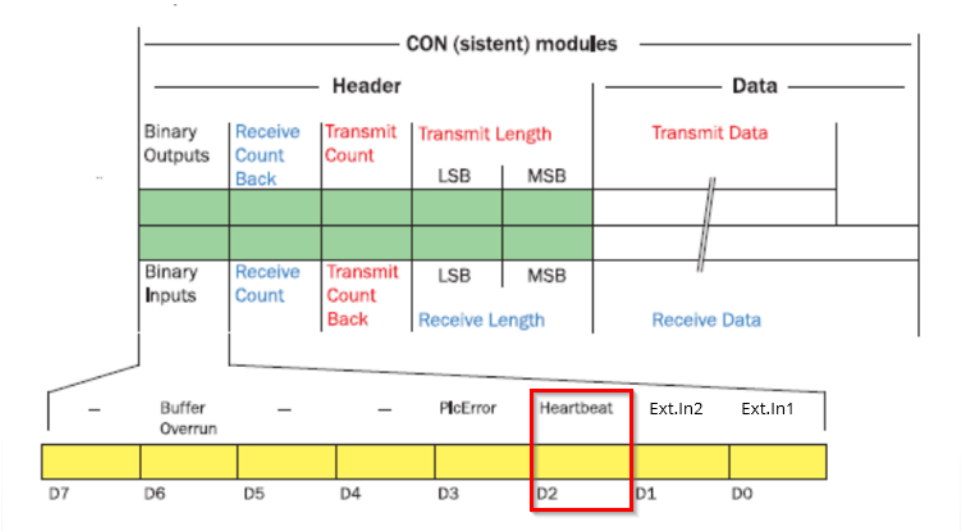
Gateway-Mode:



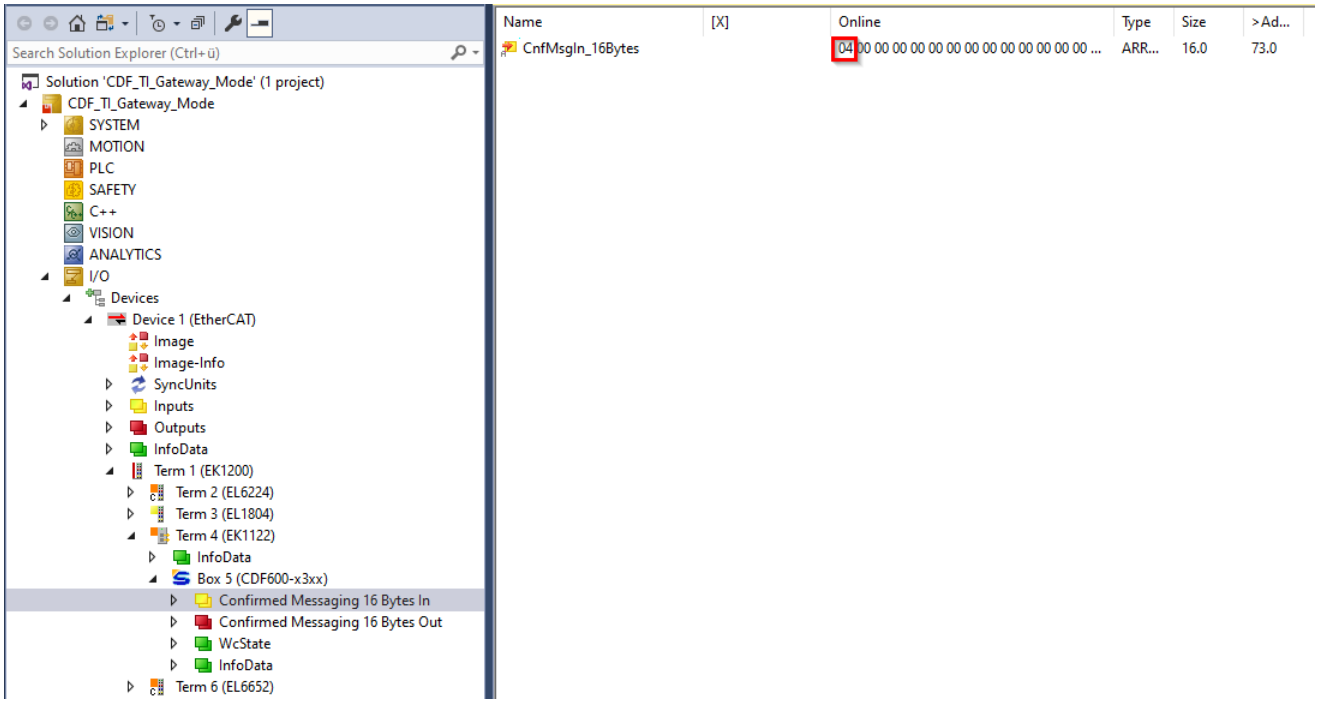
Proxy-Mode:



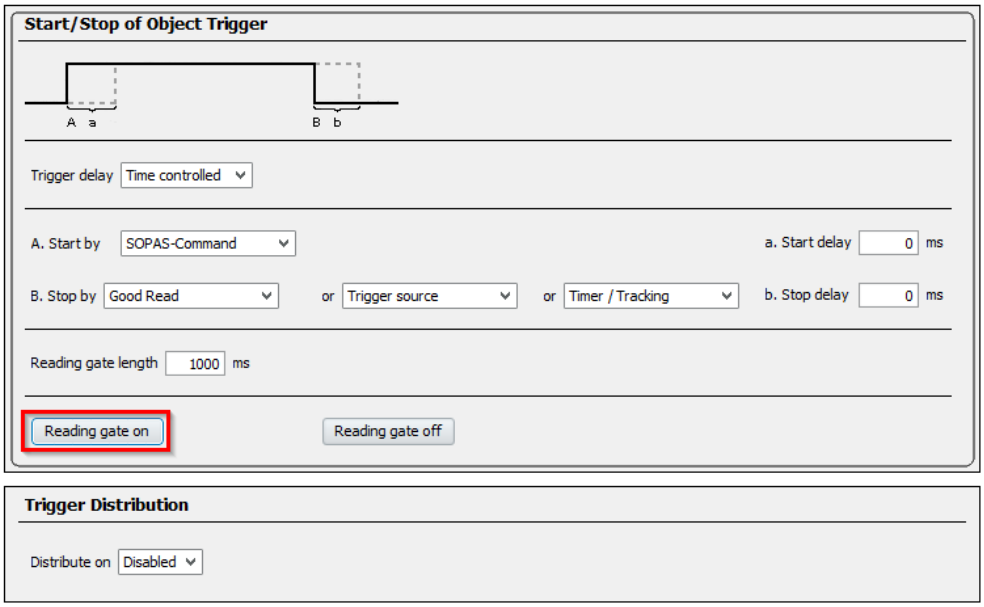
To check that there is a connection to the CDF, the heartbeat bit from the Confirmed Messaging Input Header can be looked at.



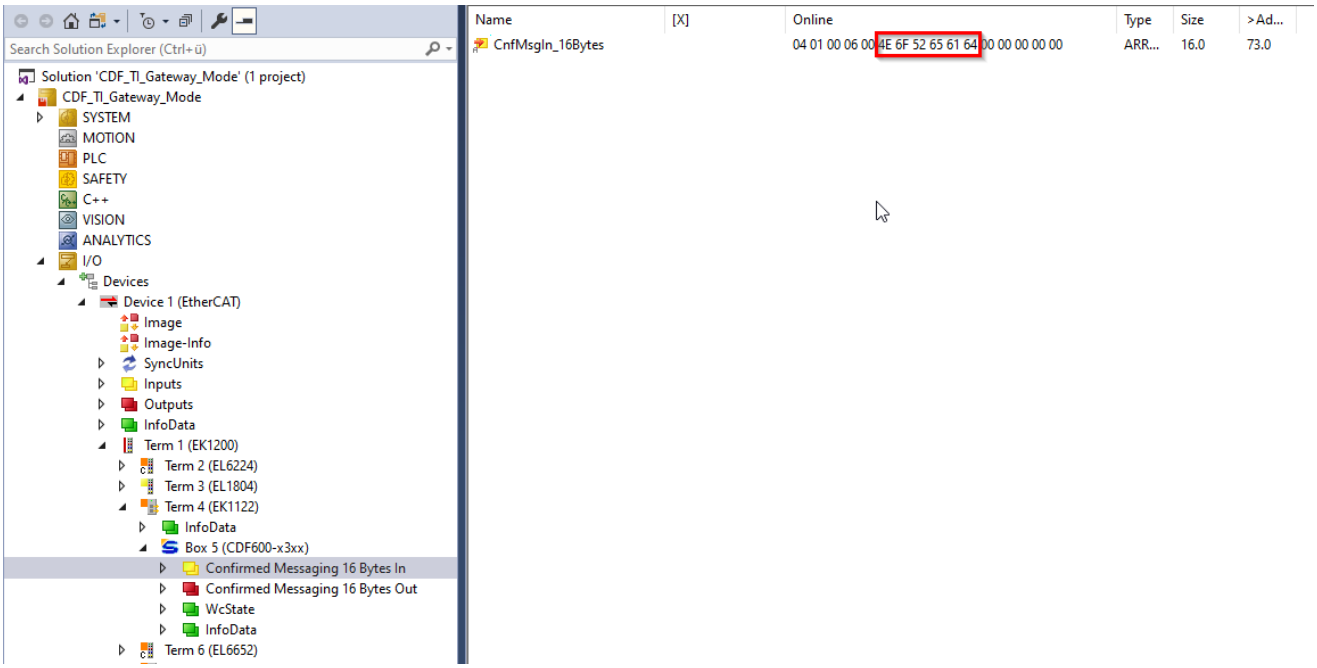
To do this, the Free Run mode can be activated in the PLC and then the first byte of the Confirmed Messaging Input should toggle between 0 and 4 (toggling of the heartbeat bit from 0 to 1).



The connected sensor can be triggered via SOPAS to check whether data from the sensor is also arriving at the PLC.



Depending on the defined output format, the PLC can then check whether the expected data has arrived. In this example, “NoRead” (Hex: 4E 6F 52 65 61 64) should be output.



6. Data exchange of the CDF600-0300

Various communication protocols can be selected for the data channel in the CDF600-0300. These modes are specific to SICK and require appropriate handling on the software side. The various communication protocols are described in the chapters below.

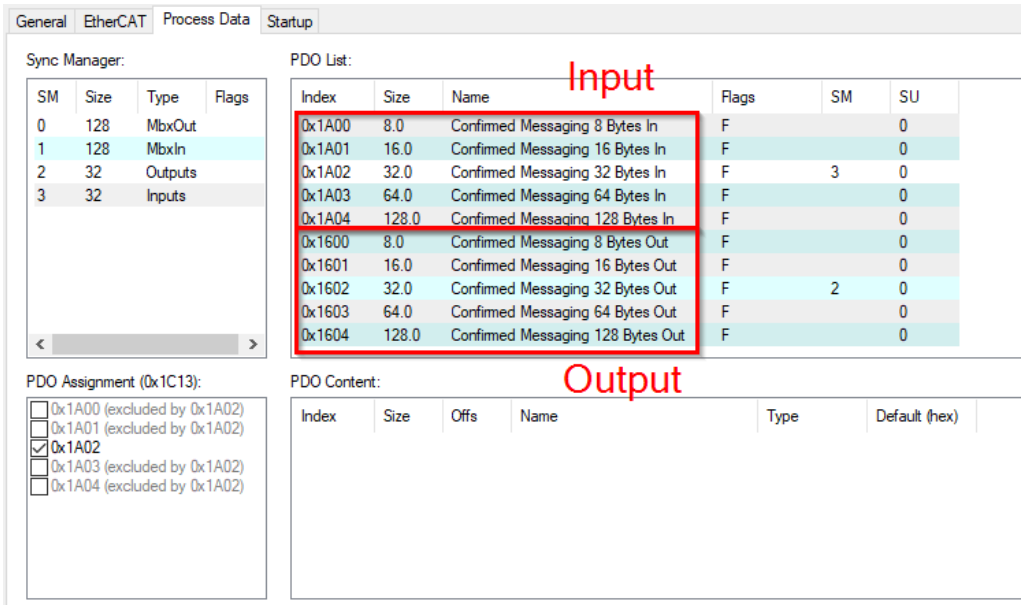
6.1. Handshake mode / Confirmed Messaging (Proxy- / Gateway-Mode)

In **Proxy-Mode**, the data channel of the CDF600-0300 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.

Max. telegram length:

Send and receive with **max. telegram length of 4000 bytes** is possible with blocking.

On PLC side a data size of up to 128 bytes (= 64 words) can be selected.



In the input/output module, 5 bytes are used for administration. A module with e.g., 32 bytes has 27 bytes of user data.

6.1.1. Byte-Layout CDF600-0300 Handshake Mode

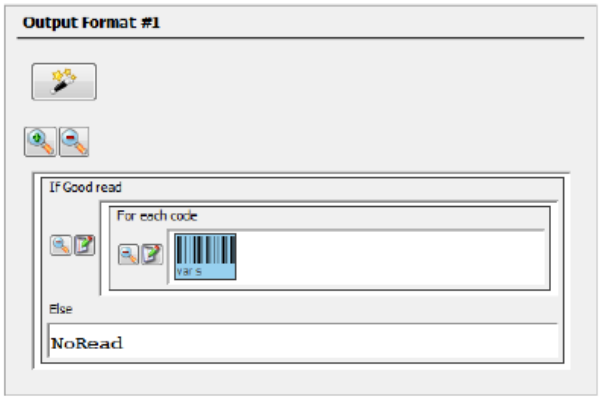
In handshake mode, the data bytes of the Confirmed Messaging look as follows. The length of the user data depends on the configured length of the Confirmed Messaging data (up to 128 bytes).

Address	**Inputs** (Data ID sensor ➔ PLC)		**Output** (Data PLC ➔ ID sensor)
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

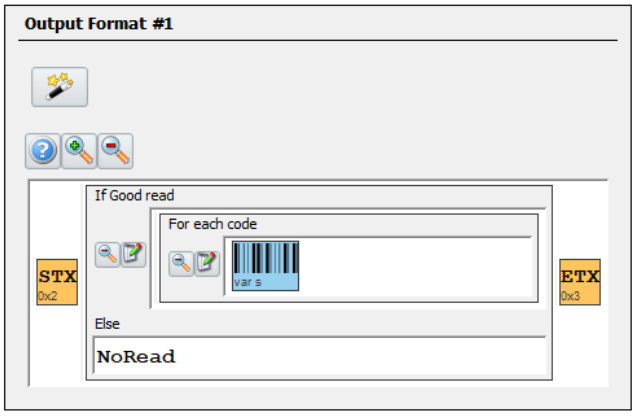
6.1.2. Receiving data Handshake Mode

Note:

Transmission from the ID sensor in **Proxy-Mode** of the CDF600-0300 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:



Transmission from the ID sensor in **Gateway-Mode** of the CDF600-0300 requires STX/ETX framing characters. The output format must be formatted **with 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 CDF600-0300 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 CDF600-0300 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 CDF600-0300 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.

6.1.3. Example 1, receipt of a single block telegram Handshake Mode

In this example, the input and output Confirmed Messaging data from the CDF have a size of 16 bytes. Two different single block telegrams (length <= 11 bytes) are received. The telegram “CLV6xx-Data” (11 bytes) is received once and then the telegram “123456789” (9 bytes) is received. As both telegrams are <= 11 bytes, no blocking is executed.

Input Byte – 16 Byte				
Bytes	Time 1	Time 2	Time 3	Time 4
Binary Input	4	4	4	4
Receive Count	1	1	2	2
Transmit Count Back	0	0	0	0
Receive Length (LSB)	11	11	9	9
Receive Length (MSB)	0	0	0	0
Data 1	„C“	„C“	„1“	„1“
Data 2	„L“	„L“	„2“	„2“
Data 3	„V“	„V“	„3“	„3“
Data 4	„6“	„6“	„4“	„4“
Data 5	„x“	„x“	„5“	„5“
Data 6	„x“	„x“	„6“	„6“
Data 7	„-“	„-“	„7“	„7“
Data 8	„D“	„D“	„8“	„8“
Data 9	„a“	„a“	„9“	„9“
Data 10	„t“	„t“		
Data 11	„a“	„a“		

Output Byte – 16 Byte				
Bytes	Time 1	Time 2	Time 3	Time 4
Binary Output	0	0	0	0
Receive Count Back	0	1	1	2
Transmit Count	0	0	0	0
Transmit Length (LSB)	0	0	0	0
Transmit Length (MSB)	0	0	0	0
Data 1				
Data 2				
Data 3				
Data 4				
Data 5				
Data 6				
Data 7				
Data 8				
Data 9				
Data 10				
Data 11				

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 CDF is now ready for the next data block.

Time 3: The next data block “123456789” (9 bytes) has been received by the CDF and the PLC shows ReceiveCount = 2.

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

6.1.4. Example 2, receipt of a blocked telegram Handshake Mode

In this example, the input and output Confirmed Messaging data from the CDF also have a size of 16 bytes. Instead of two single block telegrams one telegram that is larger than 11 Bytes is received. The content of the telegram is “CLV6xx-12345” (12 bytes). Because the telegram is larger than 11 bytes, it is received in two blocks.

Input Byte – 16 Byte			
Bytes	Time 1	Time 2/3	Time 4
Binary Input	4	4	4
Receive Count	1	2	2
Transmit Count Back	0	0	0
Receive Length (LSB)	12	1	1
Receive Length (MSB)	0	0	0
Data 1	„C“	„5“	„5“
Data 2	„L“		
Data 3	„V“		
Data 4	„6“		
Data 5	„x“		
Data 6	„x“		
Data 7	„-“		
Data 8	„1“		
Data 9	„2“		
Data 10	„3“		
Data 11	„4“		

Output Byte – 16 Byte			
Bytes	Time 1	Time 2/3	Time 4
Binary Output	0	0	0
Receive Count Back	0	1	2
Transmit Count	0	0	0
Transmit Length (LSB)	0	0	0
Transmit Length (MSB)	0	0	0
Data 1			
Data 2			
Data 3			
Data 4			
Data 5			
Data 6			
Data 7			
Data 8			
Data 9			
Data 10			
Data 11			

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 CDF 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.

6.1.5. Send data Handshake Mode

The PLC can also send data, such as commands, to the CDF600-0300 or ID sensor. In the Proxy-Mode of the CDF600-0300, 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 CDF600-0300. The send data is entered without the STX/ETX frame.

Example:

PLC output range: 4 bytes with content: **“sRI0”**

➔ 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 CDF600-0300 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 CDF600-0300 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 CDF600-0300 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 CDF600-0300 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

6.1.6. Example 3, transmission of a single block telegram Handshake Mode

In this example, two different single block telegrams are sent by the PLC. The input and output length of the Confirmed Messaging data are also 16 bytes in this example. The first telegram that is sent is “sRI0” (4 bytes) and then the telegram “sRIX” (4 bytes) is sent. The behavior is as follows:

Input Byte – 16 Byte					
Bytes	Time 1/2	Time 3/4	Time 5	Time 6/7	Time 8
Binary Input	4	4	4	4	4
Receive Count	1	2	2	3	3
Transmit Count Back	1	1	1	2	2
Receive Length (LSB)	22	11	11	6	6
Receive Length (MSB)	0	0	0	0	0
Data 1	„s“	„6“	„6“	„s“	„s“
Data 2	„R“	„2“	„2“	„F“	„F“
Data 3	„A“	„x“	„x“	„A“	„A“
Data 4	„ “	„ “	„ “	„ “	„ “
Data 5	„0“	„5“	„5“	„1“	„1“
Data 6	„ “	„ “	„ “	„1“	„1“
Data 7	„6“	„V“	„V“		
Data 8	„ “	„5“	„5“		
Data 9	„C“	„.“	„.“		
Data 10	„L“	„1“	„1“		
Data 11	„V“	„1“	„1“		

Output Byte – 16 Byte					
Bytes	Time 1/2	Time 3/4	Time 5	Time 6/7	Time 8
Binary Output	0	0	0	0	0
Receive Count Back	0	1	2	2	3
Transmit Count	1	1	1	2	2
Transmit Length (LSB)	4	4	4	4	4
Transmit Length (MSB)	0	0	0	0	0
Data 1	„s“	„s“	„s“	„s“	„s“
Data 2	„R“	„R“	„R“	„R“	„R“
Data 3	„I“	„I“	„I“	„I“	„I“
Data 4	„0“	„0“	„0“	„X“	„X“
Data 5					
Data 6					
Data 7					
Data 8					
Data 9					
Data 10					
Data 11					

Time 1: First data telegram “sRI0” (SDD version) was sent by setting TransmitCount = 1 on the output side. If the CDF 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 CDF 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 CDF 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 CDF 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 CDF is now ready for additional data.

6.1.7. Example 4, transmission of a blocked telegram Handshake Mode

In this example, a telegram larger than 11 bytes is sent by the PLC. The input and output length of the Confirmed Messaging data are also 16 bytes in this example. The telegram that is sent is “sMN mTCgateon” (13 bytes). As this telegram is larger than 11 bytes, it must be sent in two blocks.

Input Byte – 16 Byte						
Bytes	Time 1	Time 2/3	Time 4	Time 5	Time 6	Time 7
Binary Input	4	4	4	4	4	4
Receive Count	0	1	2	2	3	3
Transmit Count Back	1	2	2	2	2	2
Receive Length (LSB)	0	15	4	4	10	10
Receive Length (MSB)	0	0	0	0	0	0
Data 1		„s“	„o“	„o“	„1“	„1“
Data 2		„A“	„n“	„n“	„2“	„2“
Data 3		„N“	„ “	„ “	„3“	„3“
Data 4		„ “	„1“	„1“	„4“	„4“
Data 5		„m“			„5“	„5“
Data 6		„T“			„6“	„6“
Data 7		„C“			„7“	„7“
Data 8		„g“			„8“	„8“
Data 9		„a“			„9“	„9“
Data 10		„t“			„0“	„0“
Data 11		„e“				

Output Byte – 16 Byte						
Bytes	Time 1	Time 2/3	Time 4	Time 5	Time 6	Time 7
Binary Output	0	0	0	0	0	0
Receive Count Back	0	0	1	2	2	3
Transmit Count	1	2	2	2	2	2
Transmit Length (LSB)	13	2	2	2	2	2
Transmit Length (MSB)	0	0	0	0	0	0
Data 1	„s“	„o“	„o“	„o“	„o“	„o“
Data 2	„M“	„n“	„n“	„n“	„n“	„n“
Data 3	„N“	„N“	„N“	„N“	„N“	„N“
Data 4	„ “	„ “	„ “	„ “	„ “	„ “
Data 5	„m“	„m“	„m“	„m“	„m“	„m“
Data 6	„T“	„T“	„T“	„T“	„T“	„T“
Data 7	„C“	„C“	„C“	„C“	„C“	„C“
Data 8	„g“	„g“	„g“	„g“	„g“	„g“
Data 9	„a“	„a“	„a“	„a“	„a“	„a“
Data 10	„t“	„t“	„t“	„t“	„t“	„t“
Data 11	„e“	„e“	„e“	„e“	„e“	„e“

Time 1: The first data block of 11 bytes has been sent by setting TransmitCount = 1 with TransmitLength = 13. If the CDF 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 TransmitLength = 2. When the CDF 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.

6.1.8. Binary status bits In

The first input byte (see chapter 6.1.1) displays the binary status bits In. It contains some status bits and one heartbeat bit.

Binary status bits In Proxy-Mode:

Bit	Name	Meaning
D7	–	Reserved
D6	BufferOverrun	0: No error 1: All the output buffers in the field bus module are full. The field bus module does not accept any transmitted data when this error is present. The field bus master can repeat data transfer later stage and then, if necessary, report an error to the user.
D5	–	Reserved
D4	–	Reserved
D3	*PLC Error*	0: No error 1: The field bus module has detected 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	*Heartbeat*	0/1: Changes every second – shows the presence of the field bus module.
D1	–	Reserved

D0	–	Reserved

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

Binary status bits In Gateway-Mode:

Bit	Name	Meaning
D7	–	Reserved
D6	BufferOverrun	0: No error 1: All the output buffers in the field bus module are full. The field bus module does not accept any transmitted data when this error is present. The field bus master can repeat data transfer later stage and then, if necessary, report an error to the user.
D5	–	Reserved
D4	–	Reserved
D3	*PLC Error*	0: No error 1: The field bus module has detected 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	*Heartbeat*	0/1: Changes every second – shows the presence of the field bus module.

D1	*Ext. In 2*	In 2 input level of the field bus module.
D0	*Ext. In 1*	In 1 input level of the field bus module.

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

6.1.9. Binary status bits Out

Binary status bits Out Proxy-Mode:

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

Binary status bits Out Gateway-Mode:

Bit	Name	Meaning
D7	–	Reserved
D6	–	Reserved
D5	–	Reserved
D4	–	Reserved
D3	–	Reserved
D2	–	Reserved
D1	*Ext. Out 2*	Output on OUT 2 of the field bus module.

D0	*Ext. Out 1*	Output on OUT 1 of the field bus module.

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

6.2. No-Handshake-Mode (Proxy-Mode)

In **Proxy-Mode**, the data channel of the CDF600-0300 is operated in No-Handshake mode, when the communication protocol has been set to **CDF600 No-Handshake**.

In **Gateway-Mode** only Handshake Mode can be used.

Max. telegram length:

The max. telegram length is limited by the size of the input/output range and is **max. 123 bytes**. **There is no fragmenting / blocking**. A module may contain e.g. 128 bytes (5 bytes header, 123 bytes user data).

Recommendation:

This mode is only recommended if the length of the received data does not exceed the input range. 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.

Change mode:

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 CDF600-0300 with the attached ID sensor.

Differences to handshake-mode:

In terms of the layout, the mode is identical to the Handshake-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 in Proxy-Mode.

6.2.1. Byte layout CDF600-0300 No-Handshake Mode

The length of the data storage area is up to max. 128 bytes (123 bytes user data) 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.

Address	**Inputs** (Data ID sensor → PLC)		**Output** (Data PLC → ID sensor)
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

6.2.2. Receiving data in No-Handshake Mode

The data format is set for Proxy-Mode as in Handshake mode **without the STX/ETX frame**. The CDF600-0300 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 No-Handshake Mode

This example shows the reception of telegrams in no-handshake mode. This means that no handshake is required. The input and output length of the Confirmed Messaging data are 16 bytes. Three different telegrams are received: “12345678” (8 bytes), “SICK” (4 bytes) and “NoRead” (6 bytes). The behavior is as follows:

Input Byte – 16 Byte			
Bytes	Time 1	Time 2	Time 3
Binary Input	4	4	4
Receive Count	1	2	3
Transmit Count Back	0	0	0
Receive Length (LSB)	8	4	6
Receive Length (MSB)	0	0	0
Data 1	„1“	„S“	„N“
Data 2	„2“	„I“	„o“
Data 3	„3“	„C“	„R“
Data 4	„4“	„K“	„e“
Data 5	„5“		„a“
Data 6	„6“		„d“
Data 7	„7“		
Data 8	„8“		
Data 9			
Data 10			
Data 11			

Output Byte – 16 Byte			
Bytes	Time 1	Time 2	Time 3
Binary Output	0	0	0
Receive Count Back	0	0	0
Transmit Count	0	0	0
Transmit Length (LSB)	0	0	0
Transmit Length (MSB)	0	0	0
Data 1			
Data 2			
Data 3			
Data 4			
Data 5			
Data 6			
Data 7			
Data 8			
Data 9			
Data 10			
Data 11			

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 No-Handshake Mode

This example shows sending telegrams in no-handshake mode. Again, the confirmed messaging data is 16 bytes. The telegrams "sRI0" (4 bytes) and "sRIX" (4 bytes) are sent by the PLC.

Input Byte – 16 Byte		
Bytes	Time 1/2	Time 3/4
Binary Input	4	4
Receive Count	1	2
Transmit Count Back	1	2
Receive Length (LSB)	22	6
Receive Length (MSB)	0	0
Data 1	„s“	„s“
Data 2	„R“	„F“
Data 3	„A“	„A“
Data 4	„ “	„ “
Data 5	„0“	„1“
Data 6	„ “	„1“
Data 7	„C“	
Data 8	„L“	
Data 9	„V“	
Data 10		
Data 11		

Output Byte – 16 Byte		
Bytes	Time 1/2	Time 3/4
Binary Output	0	0
Receive Count Back	0	0
Transmit Count	1	2
Transmit Length (LSB)	4	4
Transmit Length (MSB)	0	0
Data 1	„s“	„s“
Data 2	„R“	„R“
Data 3	„I“	„I“
Data 4	„0“	„X“
Data 5		
Data 6		
Data 7		
Data 8		
Data 9		
Data 10		
Data 11		

Time 1: First data telegram “sRI0” (SDD version) was sent by setting TransmitCount = 1 on the output side. If the CDF 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 CDF 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.

7. Function of the CDF600-0300 in Gateway-Mode

This mode is active if the rotary coding switch “Mode” of the CDF600-0300 is set to 2 or 4, and then the CDF600-0300 is restarted.

In this mode, the ESI file for Gateway-Mode must be used.

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:

- The data on the serial interface must be **framed** with the control characters **STX / ETX**, if in the CDF600-0300 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.
  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 CDF600-0300 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

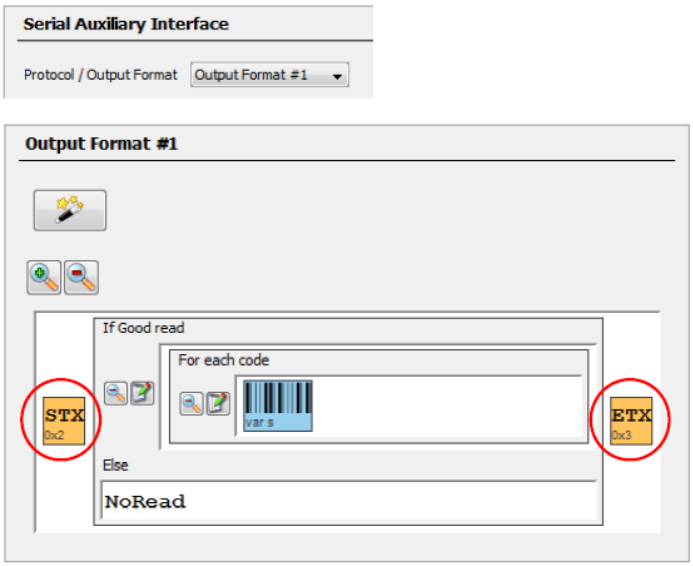
Connection of the ID sensor:

The serial interface is on pins 2, 3 and 5 in the CDF600-0300. 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**.

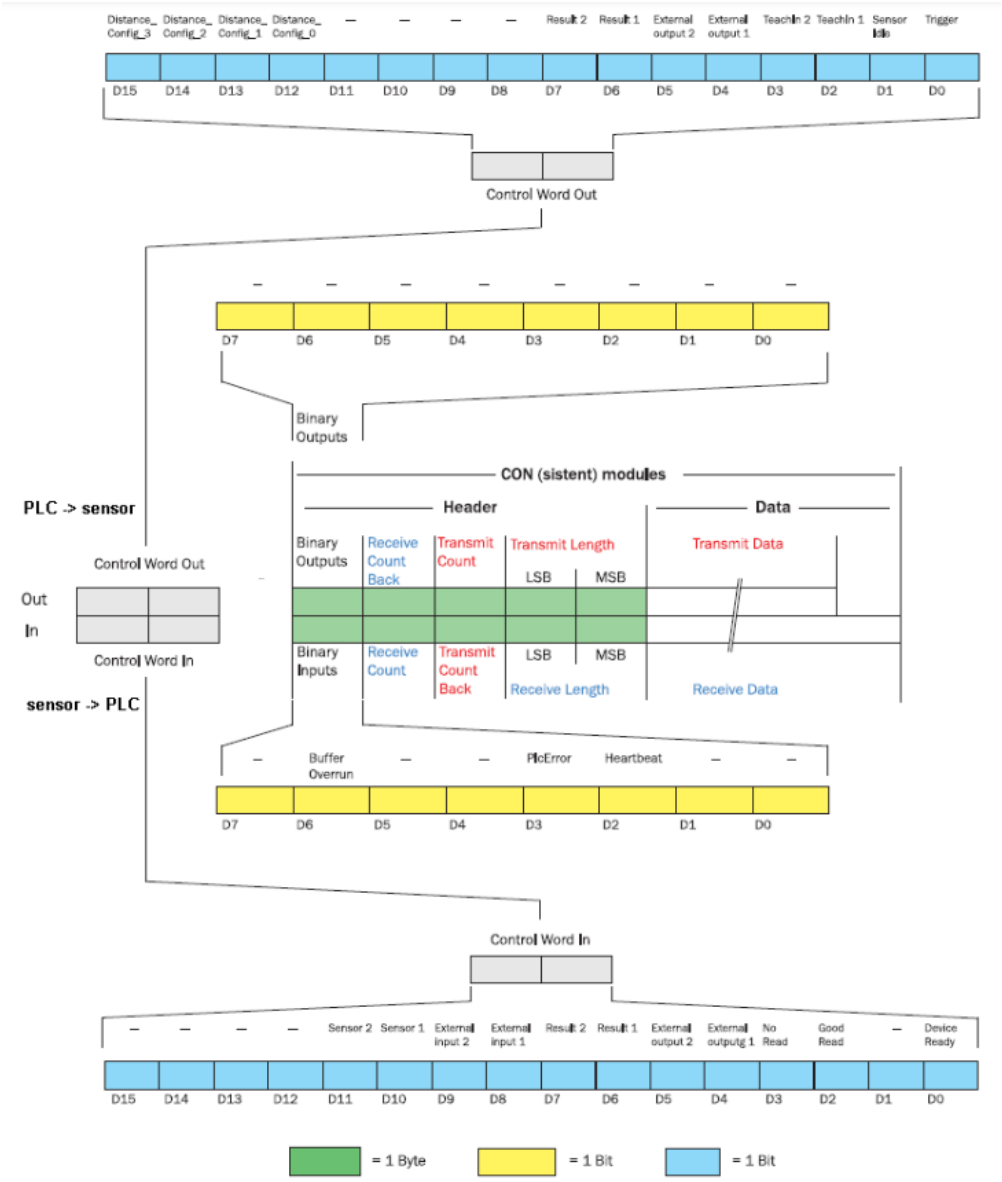
Example:

Output format 1 with STX/ETX:



8. Control bits

The control bits are only in **Proxy-Mode** available. The control bits consist of the Control Word In (16 bits) and Control Word Out (16 bits). The control bits are shown in blue in the following figure.



8.1. Control bits In

The control bits In are sent from the sensor to the PLC. The control bits In are 2 bytes long. The bits are assigned as follows:

Bit	Name	Meaning
D0	Device Ready	Status of the attached ID sensor.
D1	System Ready	Not implemented.
D2	Good Read	Status of the last read results.
D3	No Read	Status of the last read results.
D4	External Output 1	OUT 1 output level of the field bus module.
D5	External output 2	OUT 2 output level of the field bus module.
D6	Result 1	Result 1 output level of the bar code scanner.
D7	Result 2	Result 2 output level of the bar code scanner.
D8	External Input 1	IN 1 input level of the field bus module.
D9	External input 2	IN 1 input level of the field bus module.
D10	Sensor 1	Sensor 1 input level of the bar code scanner.
D11	Sensor 2	Sensor 2 input level of the bar code scanner.
D12	–	Reserved.
D13	–	Reserved.
D14	–	Reserved.
D15	–	Reserved.

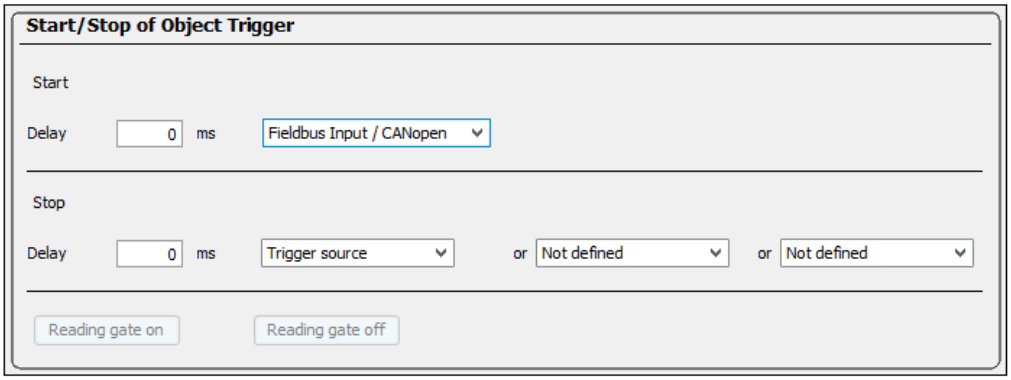
8.2. Control bits Out

The control bits Out are sent from the PLC to the sensor. The control bits Out are 2 bytes long. The bits are assigned as follows:

Bit	Name	Meaning
D0	Trigger	Object trigger for the ID sensor.
D1	Sensor Idle	The attached ID sensor is put into sleep mode.
D2	Teach In 1	The Teach In 1 teach-in process is triggered.
D3	Teach In 2	The Teach In 2 teach-in process is triggered.
D4	External Output 1	Output on OUT 1 of the field bus module.
D5	External output 2	Output on OUT 2 of the field bus module.
D6	Result 1	Output on Result 1 of the bar code scanner.
D7	Result 2	Output on Result 2 of the bar code scanner.
D8	–	Reserved
D9	–	Reserved
D10	–	Reserved
D11	–	Reserved
D12	Distance_Config_0	Controls bit 0 under dynamic (focus) switchover.
D13	Distance_Config_1	Controls bit 1 under dynamic (focus) switchover.
D14	Distance_Config_2	Controls bit 2 under dynamic (focus) switchover.
D15	Distance_Config_3	Controls bit 3 under dynamic (focus) switchover.

Example:

Set ID scanner to object trigger start via “Fieldbus Input” and object trigger stop via “Trigger Source”. The ID sensor can be triggered by the PLC using the bit D0 “Trigger”:



9. Appendix (further functions / information)

9.1. Troubleshooting

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

9.1.1. Proxy-Mode troubleshooting

- Check the proxy capability of the attached ID sensor. Can this type/version be operated in Proxy-Mode on the CDF600-0300? (See chapter 1.2) Check rotary coding switch on CDF600-0300: Mode OK?
- Switch on the CDF600-0300: Check the state of the “Power” LED (see chapter 2), the LED must light up permanently➔ ID sensor has been detected and is in operational state.
- In SOPAS, read the system information of the ID sensor via USB ➔ CDF600-0300 must have been recognized in the system status:

Information	0:03:00	0:03:00	CDF600 detected	Firmware: V1.20.32		1	0x200070A

- Is the CDF600-0300 found in the PLC? If not, check whether the correct ESI file has been used and correctly integrated. → Three different ESI files (depends on connected device and operation mode → see chapter 1).
- Check EtherCAT bus status in the PLC. The EtherCAT bus to the CDF600-0300 should be green.
- Are the states of the ECAT IN and ECAT OUT LEDs as expected (see chapter 2).

If no data arrives:

- Check HW config on PLC → are the input and output bytes linked correctly.
- Is the heartbeat bit toggeling?
- Does only the first data block appear and no others? → Then the handshake does not work correctly → Set handshake manually or with function block.

9.1.2. Gateway-Mode troubleshooting

- Check rotary switches on the CDF600-0300: Mode correct?
- Switch on the CDF600-0300: Check the state of the "Power" LED (see chapter 2), the LED must light up permanently ➔ ID sensor has been detected and is in operational state.
- Check ESI file: Is the correct ESI file being used (ESI File for Gateway-Mode → see chapter 1.3) and has the ESI file been correctly integrated?
- Check CM data: Is data coming in? Does the heartbeat bit toggle?
- Check EtherCAT bus status in the PLC. The EtherCAT bus to the CDF600-0300 should be green.
- Are the states of the ECAT IN and ECAT OUT LEDs as expected (see chapter 2).
- Was the CDF600-0300 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 **Fehler! Verweisquelle konnte nicht gefunden werden.**)
- Is the heartbeat bit toggeling?
- Does only the first data block appear and no others? → Then the handshake does not work correctly → Set handshake manually or with function block.

9.2. Resetting the CDF600-0300 to the default settings / clearing the cloning memory

In mode 2 (Gateway-Mode), the CDF600-0300 can be completely reset to the default settings. This deletes all data and the cloning parameters for the ID sensor in Proxy-Mode. This is general recommended when the CDF was previously used.

As a first step you could check, what parameter set is stored in the memory by sending the following commands (everything is STX/ETX framed):

- sRN CdfParaDevType --> checks of which device the parameter set is
- sRN CdfParaDevSwVers --> checks which software version the device of the parameter set has

Example you have a parameter set of a CLV with firmware version 5.61 stored in the cloning memory the commands and answers:

command: sRN CdfParaDevType answer: sRA CdfParaDevType E CLV622-0120
command: sRN CdfParaDevSwVers answer: sRA CdfParaDevSwVers 5 V5.61

To **delete the memory** you have to send the following three commands:

command: sMI 0 6 72173AC4 answer: sAI 0 1
command: sMN mSCloadfacdef answer: sAN mSCloadfacdef
command: sMN mEEwritepara answer: sAN mEEwritepara 1

Then **reboot** the CDF!

After that the answers should look like this:

command: sRN CdfParaDevType answer: sRA CdfParaDevType 1 -
command: sRN CdfParaDevSwVers answer: sRA CdfParaDevSwVers 1 -

Afterwards a power cycle of the CDF is needed. Then the reset of all parameter and settings is finished.

9.3. Monitoring data output via SOPAS terminal

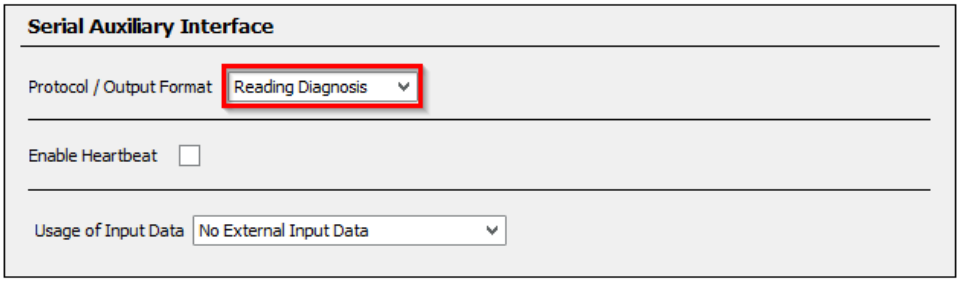
To do so, open the SOPAS terminal or another terminal program, e.g. Docklight, and establish a connection via USB or TCP/IP.

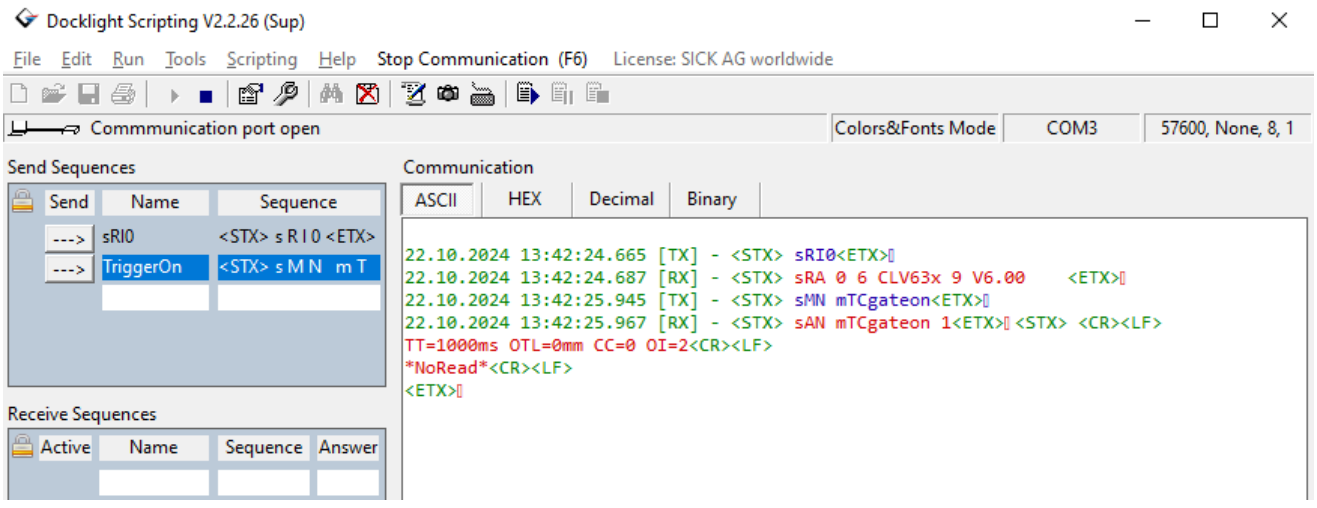
Proxy-Mode:

In Proxy-Mode, the output of the serial AUX interface 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 CDF600-0300.

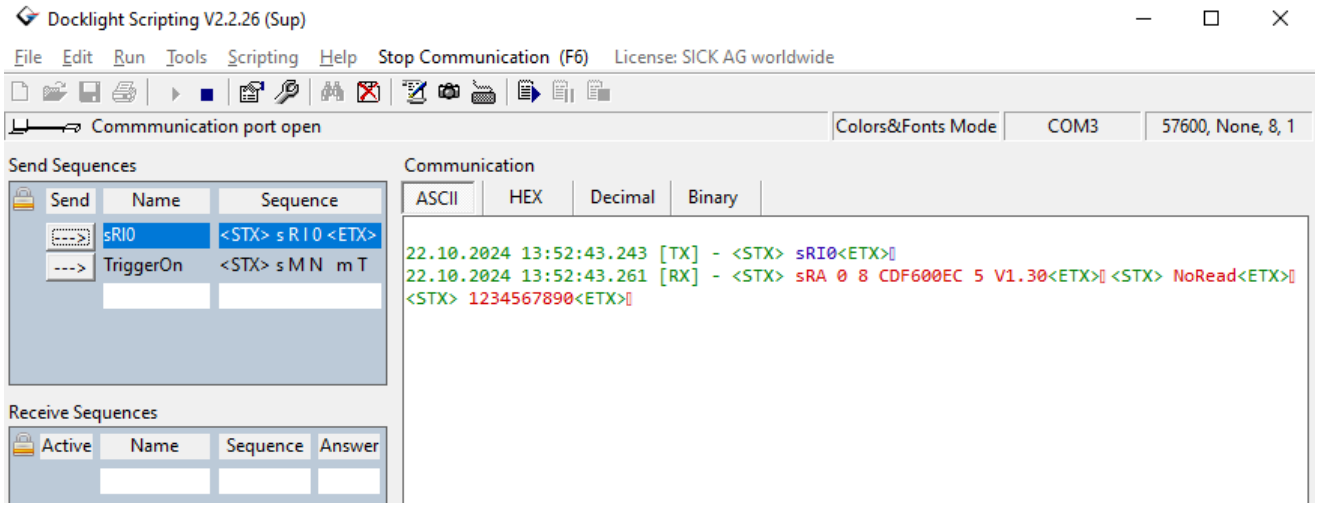
Example: reading diagnosis as serial Aux output





Gateway-Mode:

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.



9.4. Firmware update of CDF600-0300 via Packageloader

For the Update of a CDF 600-0300, you must use the package loader V2.7. **Sopas cannot be used for this update**. Time for Firmware download via serial aux is approximately 5 min.

For this update a cable connection from CDF600-0300 to your PC is necessary. Therefore, there are two cable possibilities:

- M8 to D-Sub, 9 pole cable SICK PN 6021195
- M8 to USB cable SICK PN 6034574 Please be aware that the CDF in Update mode cannot communicate with EtherCAT.

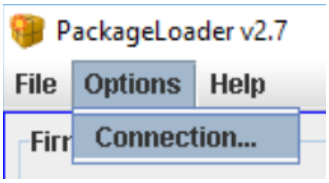
Update Process:

Set CDF600-0300 in right mode and connect to PC:

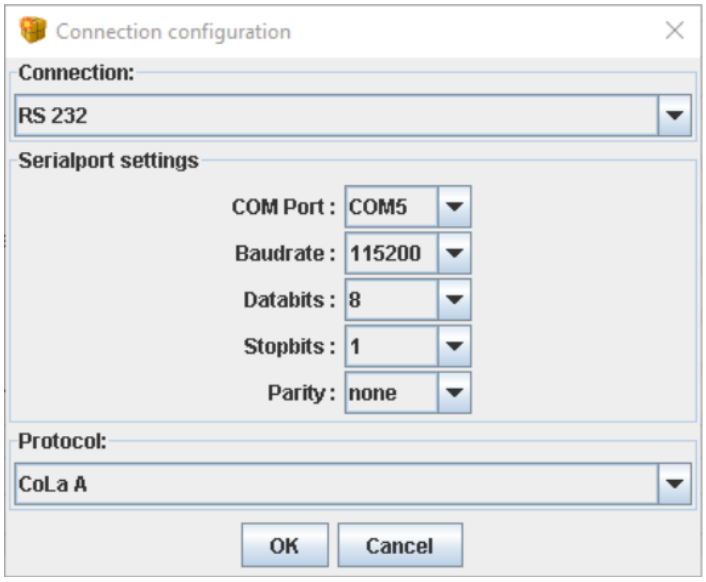
- To update the CDF600-0300 EtherCAT you have to set the Mode switch to Mode E.
- After this you need to reboot the CDF (Power Cycle) so that the CDF saves the right mode.
- If the CDF is working in this mode, there is no Bus communication possible.
- Connect the CDF afterwards with your PC (using one cable mentioned before).

Start Packageloader and connect with CDF:

- Start the Packageloader V2.7 (it only works if Packageloader was separately installed)
- Then select under Options → Connection.

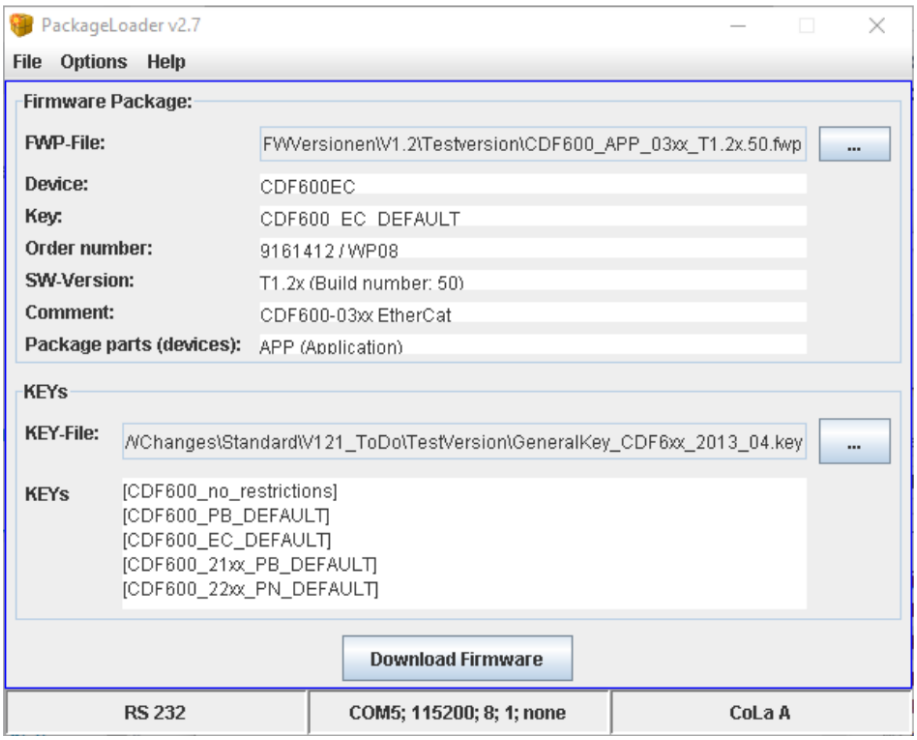


- The Connection configuration Window opens. Now the communication Parameters to connect to the CDF600-0300 have to be set.
  - **Connection:** RS232
  - Select the corresponding **COM Port** your CDF is connected to your PC.
  - **Baudrate:** 115200
  - **Databits:** 8
  - **Stopbits:** 1
  - **Parity:** none
  - The **Protocol** has to be set to CoLa A.



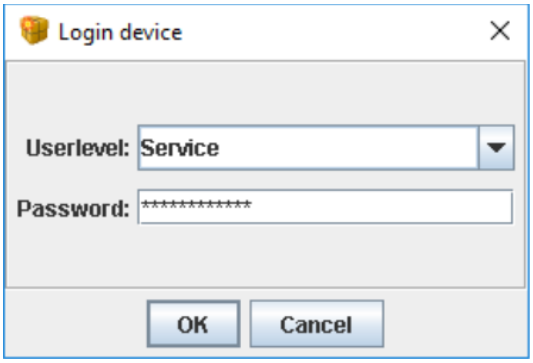
Browse for Firmware File (*.fwp) and for the keyfile (*.key)

- In the Firmware Package menu, browse for your firmware file and for the Keyfile.
- If entered, move your Window to the left side of your screen to make sure that you see all following dialogboxes which are displayed on right side.



Start Update and Login

- Start the Update by clicking the button: Download Firmware.
- The Login device window will open.
- Loging with Userlevel: *service* and Password: *servicelevel*

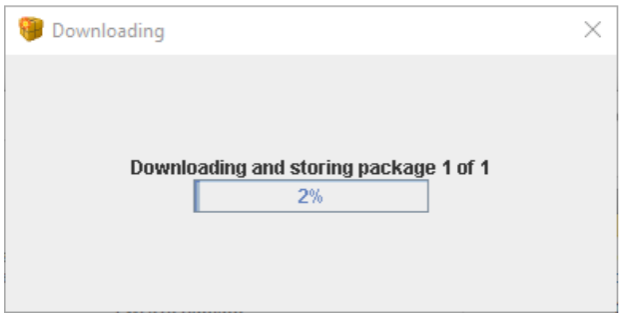


Error window

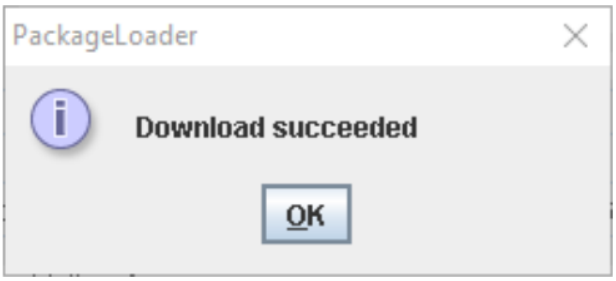
- The Error may occur during the first download.
- Please click okay and click again on the button Download Firmware.
- A new Login with Userlevel: *service* and Password: *servicelevel* will be necessary. After that the download starts.

Download Firmware

- The firmware update will need about 5 minutes.



- When the firmware update is completed, a window is opened and the device reboots.



Final Installation

After update was successful, please select the needed Mode via the Mode switch and reboot (power cycle the CDF). Check that the device boots up correctly by watching the startup message and check the device in your system.