Technical Information

SICK
CoLa Communication Block

SICK CCOM PN CP Function Block for
Siemens Step7 Controllers

SICK
Sensor Intelligence.

Table of contents

1 About this document

Please read this chapter carefully before you begin working with these operating instructions and the SICK CoLa communication block.

1.1 Purpose of this document

These operating instructions describe how to use the SICK CCOM_PN_CP function block. They are intended to guide technical personnel working for the machine manufacturer/operator through the processes of configuring and commissioning the function block.

1.2 Target group

These operating instructions are aimed at specialist personnel such as technicians and engineers.

2 General information

The CCOM_PN_CP function block facilitates data exchange between SICK devices and S7 controllers. The block supports a PROFINET connection via a Simatic CP module (communication processor). Controllers with an integrated PROFINET controller are not supported.

This block can be used to operate the following SICK sensors:
- CLV6xx
- LECTOR62x
- RFH62x
- RFU63x
- MSC800

The following figure shows how the function block is represented in the function block diagram (FBD) view.

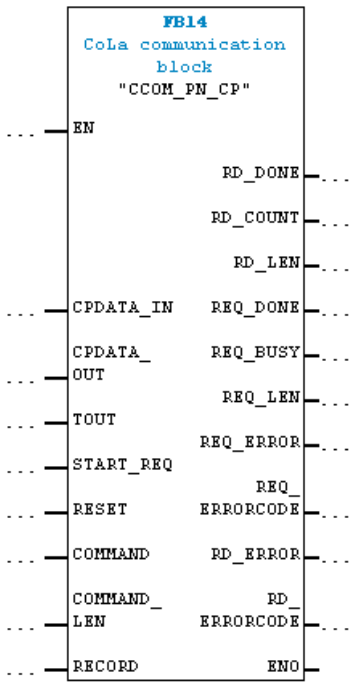

Figure 1: Representation of function block in FBD

Optional functions:
- Send CoLa[i] commands to a SICK sensor
- Receive CoLa responses from a SICK sensor
- Receive telegrams sent by devices (can be configured in SOPAS[ii] output format)

[i] The command language (CoLa) is a protocol internal to SICK for communicating with SOPAS devices.
[ii] SOPAS-ET is an engineering tool for configuring SICK sensors.

3 Hardware configuration

3.1 Supported PLC controllers

The function block may only be operated with Simatic S7-300 controllers. Only controllers that use a CP module for the PROFINET controller are supported. Controllers with integrated PROFINET controllers are not supported.

3.2 Supported fieldbus gateways/sensors

The SICK sensor communicates with the controller via PROFINET. If the sensor does not support PROFINET, a CDM425 gateway module can be used.

3.3 Configuration in Step7

The relevant sensor or gateway must be configured in the Step7 hardware configuration before the function block can be used. The first step is to import the relevant generic station description (GSDML file) into the Step7 hardware library.

The function block is specially designed for handshake mode. Only use modules from the "Handshake V2.1" category, which are defined with a length of between 8 and 128 bytes. The addresses used can be configured inside or outside of the I/O area. Addresses must not be assigned to I/O areas that have a partial process image and OB6x connection (synchronous interrupts) assigned to them.

3.3.1 Hardware configuration

Figure 2 shows a sample configuration with a CDM425 PROFINET gateway.

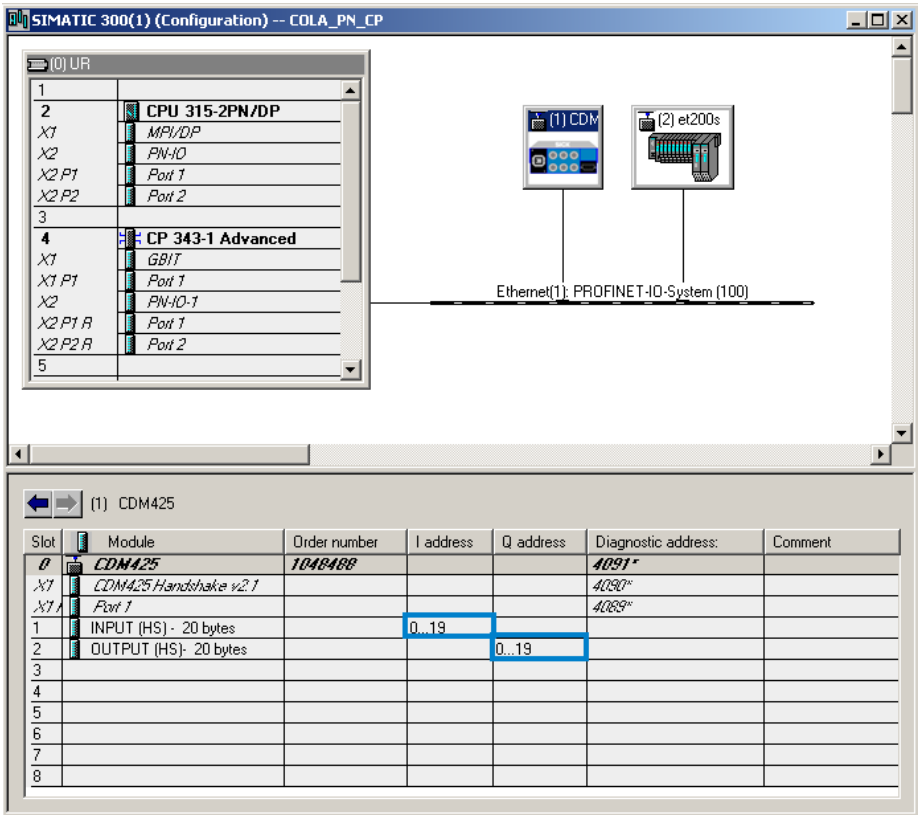

Figure 2: Step7 hardware configuration

Please note that the I/O addresses of the CP module are not identical to the I/O addresses of the CPU because the CP module has its own address range. Therefore, direct communication with the PROFINET station is not possible from the S7 program.

The entire I/O area of the CP module (in this case, the SICK CDM425 and Siemens ET200S stations) is accessed via the FC11 (PNIO_SEND) and FC12 (PNIO_RECV) functions. These functions result in a consistent I/O image of all the devices connected to the CP module.

In order to utilize the Siemens FCs, the I/O areas of the connected I/O devices must be configured in a continuous sequence, starting with address 0.

3.3.2 Access to the I/O area

The I/O area of the connected stations should be read or written to during every PLC cycle. In this example, the FC11/FC12 functions are called cyclically in OB1.

Function FC12 (PNIO_RECV) must be configured as follows:

CPLADDR:	Hardware address of the configured CP module (see Figure 4)
MODE:	0 = IO controller mode
LEN:	Byte length of all input data starting from address 0 CDM425 (0..19) = 20 bytes ET200S (20..23) = 4 bytes Total = 24 bytes
RECV:	Pointer referencing the data area (DB) where the incoming data is to be stored
IOPS:	Pointer referencing the data area (DB) where the IOPS (IO Producer Status) is to be stored. The data area for the IOPS data must have a length of LEN/8 (24/8 = 3 bytes).

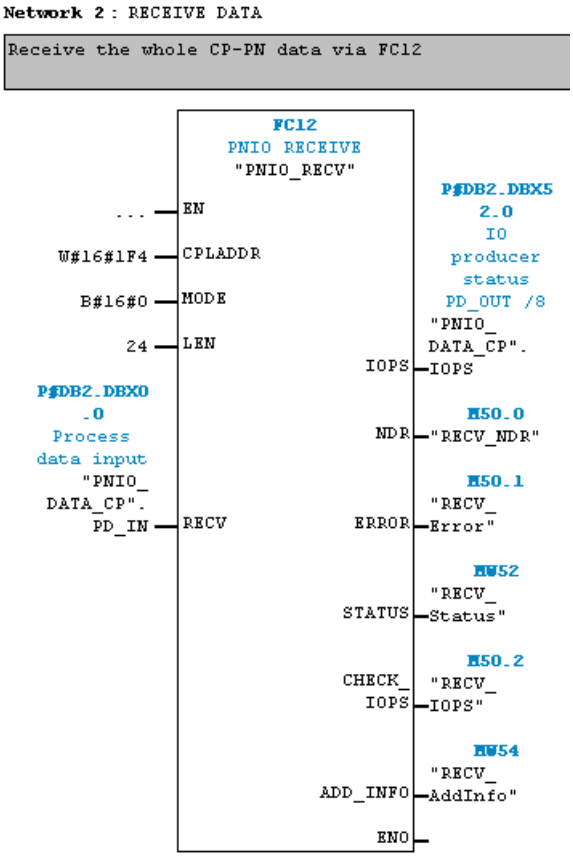

Figure 3: FC12 (PNIO_RECV) call in OB1

Figure 4 shows you where to find the hardware address of the configured CP module in the Simatic hardware configuration.

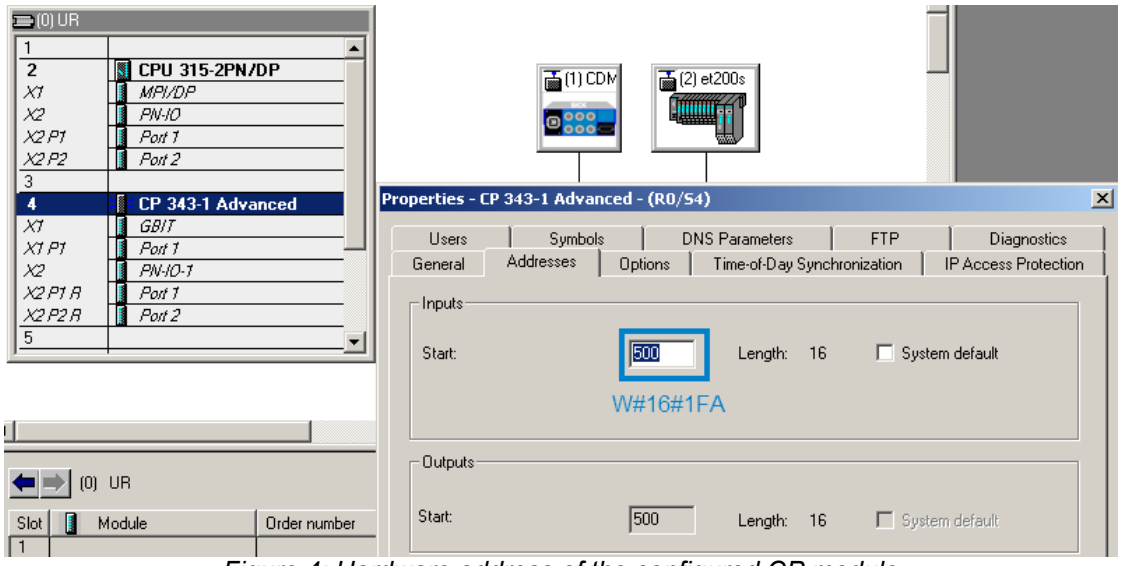

Figure 4: Hardware address of the configured CP module

Function FC11 (PNIO_SEND) must be configured as follows:

CPLADDR:	Hardware address of the configured CP module (see Figure 4)
MODE:	0 = IO controller mode
LEN:	Byte length of all output data starting from address 0 CDM425 (0..19) = 20 bytes ET200S (20..23) = 4 bytes Total = 24 bytes
SEND:	Pointer referencing the data area (DB) used to store the output data that is to be written
IOCS:	Pointer referencing the data area (DB) where the IOCS (IO Consumer Status) is to be stored. The data area for the IOCS data must have a length of LEN/8 (24/8 = 3 bytes).

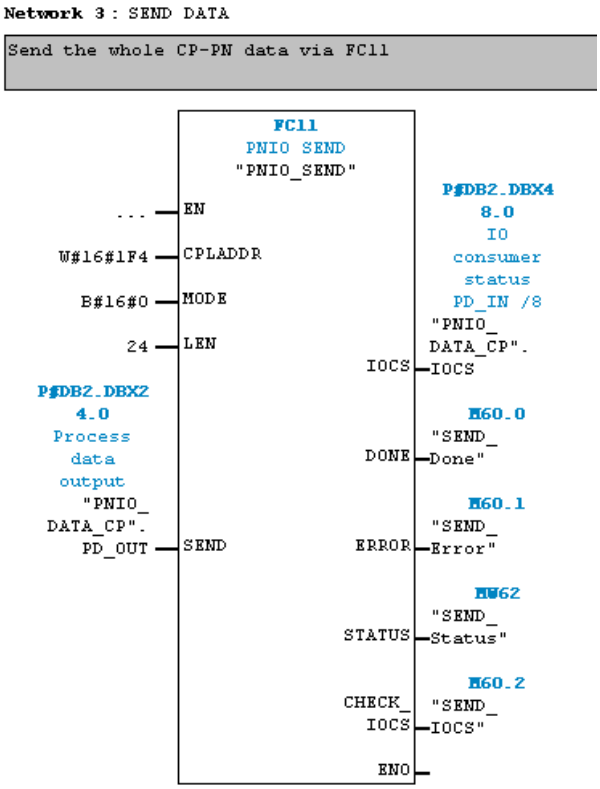

Figure 5: FC11 (PNIO_SEND) call in OB1

4 Block description

The CCOM_PN_CP (FB14) function block makes it easier to use SICK sensors with S7 controllers. The block enables you to send and receive CoLa telegrams via a PROFINET connection that has been set up in the hardware configuration.

The block can be used for the following tasks:
- Send CoLa commands to a SICK sensor
- Receive CoLa responses from a SICK sensor
- Receive telegrams sent by devices (can be configured in SOPAS output format)

The block automatically fragments the data as soon as it cannot be transmitted/received in a cycle.

The function block is an asynchronous FB, i.e., processing encompasses several function block calls. Therefore, the function block must be called cyclically in the user program.

4.1 Block specifications

Block number:	FB14
Block name:	CCOM_PN_CP
Version:	1.0
Blocks called:	SFC20 (BLKMOV) SFB4 (TON)
Data blocks used:	-
Block call:	Cyclical
Flags used:	None
Counters used:	None
Registers used:	AR1, AR2 (for multi-instances)
Language used for block creation:	Step7 STL

The system blocks (SFCs/SFBs) used in the function block must exist on the controller that is being used.

4.2 Operating principle

The following parameters must be specified before the CCOM_PN_CP block can be used.

CPDATA_IN: Pointer referencing the input data of the sensor/gateway. The input data first has to be fetched using function FC12 (PNIO_RECV).

CPDATA_OUT: Pointer referencing the output data of the sensor/gateway. The output data has to be transmitted to the device using function FC11 (PNIO_SEND).

COMMAND: The pointer references the data area in which the CoLa command is stored. The data area must be created by the programmer (e.g., data block with an array of CHAR). The command must be specified without [STX]/[ETX] framing.

COMMAND_LEN: Character length of the CoLa command to be transmitted

RECORD: The pointer references the data area in which the telegrams sent by the device are stored. The data area must be created by the programmer (e.g., data block with an array of BYTE).

4.2.1 Receiving reading results (RD)

Data sent by the device (RD) is written to the record as soon as the function block receives new data. For one PLC cycle, the RD_DONE bit indicates that new data has been received. The RD_COUNT counter is incremented as soon as new data has been received. The RD_LEN parameter indicates the byte length of the telegram last received.

4.2.2 Device communication via CoLa commands (REQ)

When communication takes place via CoLa commands, the command defined in COMMAND is transmitted to the device. The resulting response is stored in the area defined by the RECORD pointer.

To start transmission, you must trigger the START_REQ parameter with a rising edge. Until a valid response is received in reply to the CoLa command sent, the REQ_BUSY parameter signals that a response is still pending. If no response is received within the timeout period (TOUT), the function is terminated with a timeout error (REQ_ERRORCODE). The REQ_DONE output parameter indicates that a response to a CoLa command has been received (REQ_DONE = TRUE).

4.2.3 Timing

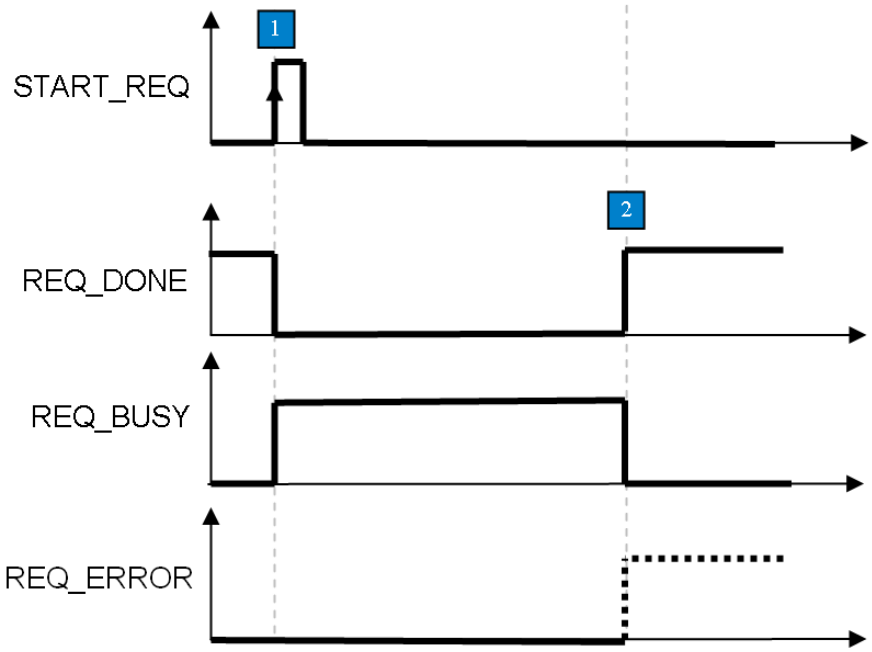

Figure 6: Timing diagram

1: Request triggered by rising edge at START_REQ. The CoLa command referenced by the COMMAND parameter is sent to the sensor. Only one command can be sent at a time.

2: Once the command has been sent and the response received, the function is terminated with "REQ_DONE". If an error occurred during the function, the function is terminated with "REQ_ERROR". "REQ_ERRORCODE" contains the error code that occurred if the function is aborted with "REQ_ERROR".

4.3 Response to errors

The REQ_ERROR or RD_ERROR bits signal that an error has occurred. In this case, an error code is output via the REQ_ERRORCODE or RD_ERRORCODE parameters. The REQ_ERROR bit remains set until a new command is started. The RD_ERROR parameter is only ever active for one PLC cycle and is then reset unless the error remains.

4.4 Resetting communication

The RESET bit can be used to reset communication between the gateway/sensor and the PLC. This involves initializing the first eight bytes of the I/O output side with zero for one second. The reset command is executed as soon as RESET = TRUE and START_REQ is triggered with a rising edge. The REQ_BUSY bit signals that the command is being processed. Once the reset routine is completed, the REQ_DONE bit is set.

4.5 Parameters

Parameter	Declaration	Data type	Memory area	Description
EN	INPUT	BOOL	I,M,D,L, const.	Enable
CPDATA_IN	INPUT	ANY	D	Pointer referencing the input area of the sensor/gateway. Only the BYTE data type is permitted. Note: Please be aware that the DB parameter data always has to be specified in its entirety for the parameter (e.g.: P#DB13.DBX0.0 BYTE 100). An explicit DB number cannot be omitted; otherwise a block error will occur.
CPDATA_ OUT	INPUT	ANY	D	Pointer referencing the output area of the sensor/gateway. Only the BYTE data type is permitted. Note: Please be aware that the DB parameter data always has to be specified in its entirety for the parameter (e.g.: P#DB13.DBX0.0 BYTE 100). An explicit DB number cannot be omitted; otherwise a block error will occur.
TOUT	INPUT	TIME	I,M,D,L, const.	Period of time, after which a timeout error is triggered. If this parameter is not connected, the timeout period is set to 5 seconds by default. Please note that some CoLa commands require longer processing periods (e.g., save commands).
START_REQ	INPUT	BOOL	I,M,D,L	Rising edge: System sends CoLa command and waits for corresponding response
RESET	INPUT	BOOL	I,M,D,L, const.	Communication is reset (HS counter of data flow protocol).

Parameter	Declaration	Data type	Memory area	Description
COMMAND	INPUT	ANY	D	Pointer referencing the area containing the CoLa command to be sent. Only the BYTE data type is permitted. The command must be specified without [STX]/[ETX] framing. Note: Please be aware that the DB parameter data always has to be specified in its entirety for the parameter (e.g.: P#DB13.DBX0.0 BYTE 100). An explicit DB number cannot be omitted; otherwise a block error will occur.
COMMAND_ LEN	INPUT	INT	I,M,D,L, const.	Number of bytes in the CoLa command to be sent, which is referenced by the #COMMAND pointer
RECORD	INPUT	ANY	D	Pointer referencing the area in which the telegrams sent by the device are stored. Only the BYTE data type is permitted. Note: Please be aware that the DB parameter data always has to be specified in its entirety for the parameter (e.g.: P#DB13.DBX0.0 BYTE 100). An explicit DB number cannot be omitted; otherwise a block error will occur.
RD_DONE	OUTPUT	BOOL	Q,M,D,L	Rising edge: A reading result sent by the device has been received (for formatting details, see SOPAS output format). Whenever a reading result is received, the bit is set for one PLC cycle. The reading result is available in the memory area referenced by the #RECORD parameter.
RD_ COUNT	OUTPUT	BYTE	Q,M,D,L	Counts the number of reading results received. The counter goes from 0 to 255 (decimal). The counter restarts at 0 once 255 has been exceeded.
RD_ LEN	OUTPUT	INT	Q,M,D,L	Indicates the byte length of the reading result received

Parameter	Declaration	Data type	Memory area	Description
REQ_DONE	OUTPUT	BOOL	Q,M,D,L	Indicates whether a CoLa command has been sent and a response received TRUE: Successfully completed FALSE: Not yet completed The command response is available in the memory area referenced by the #RECORD parameter.
REQ_BUSY	OUTPUT	BOOL	Q,M,D,L	REQ command in progress
REQ_ LEN	OUTPUT	INT	Q,M,D,L	Length of a response telegram in BYTES
REQ_ ERROR	OUTPUT	BOOL	Q,M,D,L	REQ error status: 0: No error 1: Aborted with error
RD_ERROR	OUTPUT	BOOL	Q,M,D,L	RD error status: 0: No error 1: Aborted with error
REQ_ERRO RCODE	OUTPUT	WORD	Q,M,D,L	REQ error status (see "Error codes")
RD_ERROR CODE	OUTPUT	WORD	Q,M,D,L	RD error status (see "Error codes")
ENO	OUTPUT	BOOL	Q,M,D,L	Enable output (LD and FBD)

4.6 Error codes

The REQ_ERRORCODE and RD_ERRORCODE parameters contain the following error information:

Error code	Brief description	Description
W#16#0000	No error	No error
W#16#0001	Invalid memory area specified for #RECORD pointer	Invalid memory area for specified ANY pointer. A DB must be assigned to the pointer.
W#16#0002	Invalid pointer length specified for #RECORD pointer	The referenced data block is shorter than the length defined by the pointer.
W#16#0003	Invalid memory area specified for #COMMAND pointer	Invalid memory area for specified ANY pointer. A DB must be assigned to the pointer.
W#16#0004	Invalid pointer length specified for #COMMAND pointer	The referenced data block is shorter than the length defined by the pointer.
W#16#0005	Timeout	The command could not be executed within the defined timeout period. Possible causes: - Device is not connected to the PLC - Incorrect communication parameters - CoLa commands have been used that do not send back responses (echo). - Command processing time > timeout period
W#16#0006	Invalid command length	The command being sent is longer than the specified command length (COMMAND_LEN).
W#16#0007	SFC20 error	The SFC20 (BLKMOV) block is signaling a block error. The error code is indicated in variable "nStatusBLKMOV" of the instantiated data block. To interpret the error code, please refer to the Step7 help system.
W#16#000A	Received telegram > #RECORD length	The received telegram is longer than the specified #RECORD length.
W#16#000B	Invalid input module	The length configured for the input module is invalid (CPDATA_IN pointer length). Valid value range: [8..128]
W#16#000C	Invalid output module	The length configured for the output module is invalid (CPDATA_OUT pointer length). Valid value range: [8..128]
W#16#000D	Internal block error	Internal block error

5 Example

Figure 7 shows an example of a connected CCOM_PN_CP function block. A SICK device with a process data width of 20 bytes input/output has been set up in the hardware configuration. In addition to the SICK device, another device has also been configured on the CP module. The I/O areas of both devices are written to/read using functions FC11/FC12 (see chapter 3).

Program call:

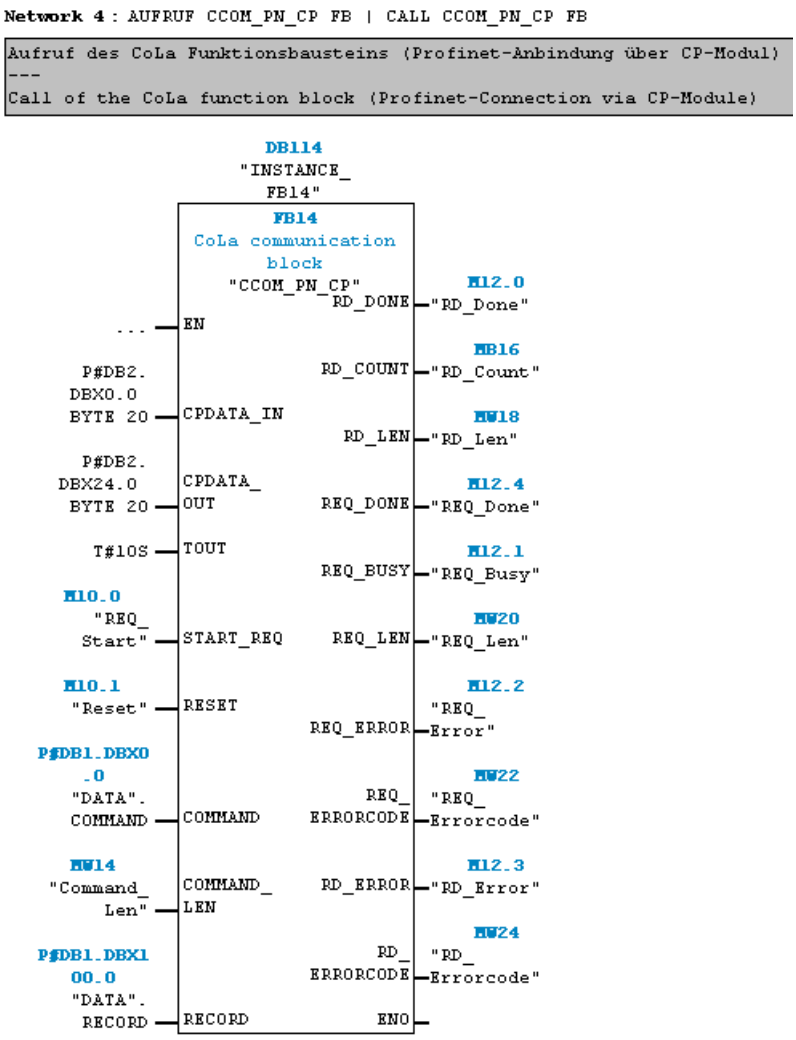

Figure 7: CCOM_PN_CP function block call in OB1

Table of variables for executing a CoLa command:

// CCOM PN CP function block				
M 10.0	"REQ_Start"	BOOL	true	
M 10.1	"Reset"	BOOL	false	
// Reading Result Status				
M 12.0	"RD_Done"	BOOL	false	
MB 16	"RD_Count"	DEZ	101	
M 12.3	"RD_Error"	BOOL	false	
MW 24	"RD_Errorcode"	HEX	W#16#0000	
// Requesting Result Status				
M 12.4	"REQ_Done"	BOOL	true	
M 12.1	"REQ_Busy"	BOOL	false	
M 12.2	"REQ_Error"	BOOL	false	
MW 22	"REQ_Errorcode"	HEX	W#16#0000	
// Command				
MW 14	"Command_Len"	DEZ	13	13
DB1.DBB 0	"DATA".COMMAND[1]	ZEICHEN	's'	's'
DB1.DBB 1	"DATA".COMMAND[2]	ZEICHEN	'M'	'M'
DB1.DBB 2	"DATA".COMMAND[3]	ZEICHEN	'N'	'N'
DB1.DBB 3	"DATA".COMMAND[4]	ZEICHEN	' '	' '
DB1.DBB 4	"DATA".COMMAND[5]	ZEICHEN	'm'	'm'
DB1.DBB 5	"DATA".COMMAND[6]	ZEICHEN	'T'	'T'
DB1.DBB 6	"DATA".COMMAND[7]	ZEICHEN	'C'	'C'
DB1.DBB 7	"DATA".COMMAND[8]	ZEICHEN	'g'	'g'
DB1.DBB 8	"DATA".COMMAND[9]	ZEICHEN	'a'	'a'
DB1.DBB 9	"DATA".COMMAND[10]	ZEICHEN	't'	't'
DB1.DBB 10	"DATA".COMMAND[11]	ZEICHEN	'e'	'e'
DB1.DBB 11	"DATA".COMMAND[12]	ZEICHEN	'o'	'o'
DB1.DBB 12	"DATA".COMMAND[13]	ZEICHEN	'n'	'n'
DB1.DBB 13	"DATA".COMMAND[14]	ZEICHEN	B#16#00	
DB1.DBB 14	"DATA".COMMAND[15]	HEX	B#16#00	
DB1.DBB 15	"DATA".COMMAND[16]	ZEICHEN	B#16#00	
DB1.DBB 16	"DATA".COMMAND[17]	ZEICHEN	B#16#00	
DB1.DBB 17	"DATA".COMMAND[18]	ZEICHEN	B#16#00	
DB1.DBB 18	"DATA".COMMAND[19]	ZEICHEN	B#16#00	
DB1.DBB 19	"DATA".COMMAND[20]	ZEICHEN	B#16#00	

The CoLa command ("sMN mTCgateon" in this case) is executed as soon as the "REQ_START" bit is triggered with a rising edge. The length of the command is transferred to the Command_Len parameter (in this case: 13 characters).

Table of variables for incoming command responses:

// Record

MW 18	"RD_Len"	DEZ	31
MW 20	"REQ_Len"	DEZ	15
DB1.DBB 100	"DATA".RECORD[1]	ZEICHEN	's'
DB1.DBB 101	"DATA".RECORD[2]	ZEICHEN	'A'
DB1.DBB 102	"DATA".RECORD[3]	ZEICHEN	'N'
DB1.DBB 103	"DATA".RECORD[4]	ZEICHEN	' '
DB1.DBB 104	"DATA".RECORD[5]	ZEICHEN	'm'
DB1.DBB 105	"DATA".RECORD[6]	ZEICHEN	'T'
DB1.DBB 106	"DATA".RECORD[7]	ZEICHEN	'C'
DB1.DBB 107	"DATA".RECORD[8]	ZEICHEN	'g'
DB1.DBB 108	"DATA".RECORD[9]	ZEICHEN	'a'
DB1.DBB 109	"DATA".RECORD[10]	ZEICHEN	't'
DB1.DBB 110	"DATA".RECORD[11]	ZEICHEN	'e'
DB1.DBB 111	"DATA".RECORD[12]	ZEICHEN	'o'
DB1.DBB 112	"DATA".RECORD[13]	ZEICHEN	'n'
DB1.DBB 113	"DATA".RECORD[14]	ZEICHEN	' '
DB1.DBB 114	"DATA".RECORD[15]	ZEICHEN	'1'
DB1.DBB 115	"DATA".RECORD[16]	ZEICHEN	B#16#00
DB1.DBB 116	"DATA".RECORD[17]	ZEICHEN	B#16#00
DB1.DBB 117	"DATA".RECORD[18]	ZEICHEN	B#16#00
DB1.DBB 118	"DATA".RECORD[19]	ZEICHEN	B#16#00
DB1.DBB 119	"DATA".RECORD[20]	ZEICHEN	B#16#00

The response (REQ) to a sent command (in this case: "sAN mTCgateon 1") becomes available in the record area as soon as the value of the "REQ_DONE" output bit changes from FALSE to TRUE (rising edge). The "REQ_LEN" parameter indicates how many bytes were received and are valid.

Table of variables for incoming reading results:

// Record			
MW 18	"RD_Len"	DEZ	9
MW 20	"REQ_Len"	DEZ	15
DB1.DBB 100	"DATA".RECORD[1]	ZEICHEN	'1'
DB1.DBB 101	"DATA".RECORD[2]	ZEICHEN	'2'
DB1.DBB 102	"DATA".RECORD[3]	ZEICHEN	'3'
DB1.DBB 103	"DATA".RECORD[4]	ZEICHEN	'4'
DB1.DBB 104	"DATA".RECORD[5]	ZEICHEN	'5'
DB1.DBB 105	"DATA".RECORD[6]	ZEICHEN	'6'
DB1.DBB 106	"DATA".RECORD[7]	ZEICHEN	'7'
DB1.DBB 107	"DATA".RECORD[8]	ZEICHEN	'8'
DB1.DBB 108	"DATA".RECORD[9]	ZEICHEN	'9'
DB1.DBB 109	"DATA".RECORD[10]	ZEICHEN	't'
DB1.DBB 110	"DATA".RECORD[11]	ZEICHEN	'e'
DB1.DBB 111	"DATA".RECORD[12]	ZEICHEN	'o'
DB1.DBB 112	"DATA".RECORD[13]	ZEICHEN	'n'
DB1.DBB 113	"DATA".RECORD[14]	ZEICHEN	' '
DB1.DBB 114	"DATA".RECORD[15]	ZEICHEN	'1'
DB1.DBB 115	"DATA".RECORD[16]	ZEICHEN	B#16#00
DB1.DBB 116	"DATA".RECORD[17]	ZEICHEN	B#16#00
DB1.DBB 117	"DATA".RECORD[18]	ZEICHEN	B#16#00
DB1.DBB 118	"DATA".RECORD[19]	ZEICHEN	B#16#00
DB1.DBB 119	"DATA".RECORD[20]	ZEICHEN	B#16#00

Data sent by the device (RD) is written to the record as soon as the function block receives new data. For one PLC cycle, the "RD_DONE" bit indicates that new data has been received (signal changes from FALSE to TRUE). The RD_COUNT counter is incremented as soon as new data has been received. The "RD_LEN" parameter indicates how many bytes were received and are valid.