Technical Information

SICK
CoLa Communication Module

CCOM_PNDP Function block for
Siemens Step7 PLC controls

SICK
Sensor Intelligence.

Table of Contents

1 About This Document

Please read these chapters carefully before you begin working with these operating instructions and the CoLa communication module.

1.1 Function of This Document

These operating instructions describe how to use the SICK CCOM_PNDP function module. They are used to guide technical personnel working for the machine manufacturer/operator in project planning and commissioning the function module.

1.2 Target Group

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

2 General Information

The CCOM_PNDP function module supports the data exchange between SICK devices and S7-Controllers. Only S7-300/S7-400 controllers are supported, which have an integrated field bus interface (Profibus / Profinet).

The following SICK sensors can be controlled by this module:
- CLV6xx
- LECTOR62x
- RFH62x
- RFU63x
- MSC800

The following figures show the representation of the function module in the function block diagram (FBD).

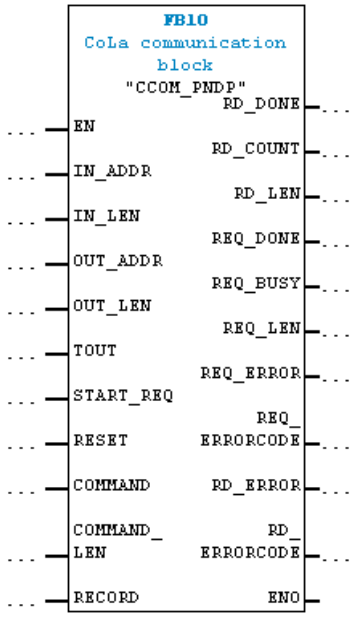

Figure 1: Representation of the function module in FUP

Optional Functions:
- Sending of CoLa[i] commands to a SICK sensor
- Receiving of CoLa answers from a SICK sensor
- Receiving of device transmitted telegrams (in SOPAS[ii] output format configurable)

[i] The command language (CoLa) is a SICK internal protocol for communication with SOPAS devices
[ii] SOPAS-ET is an Engineering Tool for the parameterizing of SICK sensors

3 Hardware Configuration

3.1 Supported SPS-Controllers

The function module may only be operated with Simatic S7 300 and 400 controllers. Only controllers with directly integrated field bus interface are supported. Communication via a communication processor (CP module) is not supported.

3.2 Supported field bus Gateways / Sensors

The SICK sensor communicates via a fieldbus (Profibus/Profinet) with the controller. If the sensor does not support the above-named fieldbuses a gateways modules can be applied.

The function module supports the following gateways:

- CDM 425 (Profinet)
- CDF 600 (Profibus/Profinet)
- CDM 420 + CMF400 (Profibus)

3.3 Configuration in Step7

The corresponding sensor respectively gateway must be configured properly in the Step7 hardware configuration before the function module can be used. The first step is to import the correct device description (GSD / GSDML) into the Step7 hardware library.

The function block is laid out especially for the handshake mode. Please do only use HS-Modules with a length between 8…128 Bytes. The used addresses can be projected in the periphery or outside. An address assignment on the periphery to which a partly process image with OB6x-connection (alarm of asynchronous trigger) is assigned, must not be used.

Figure 2 shows a sample configuration with a CDM-425 Profinet Gateway.

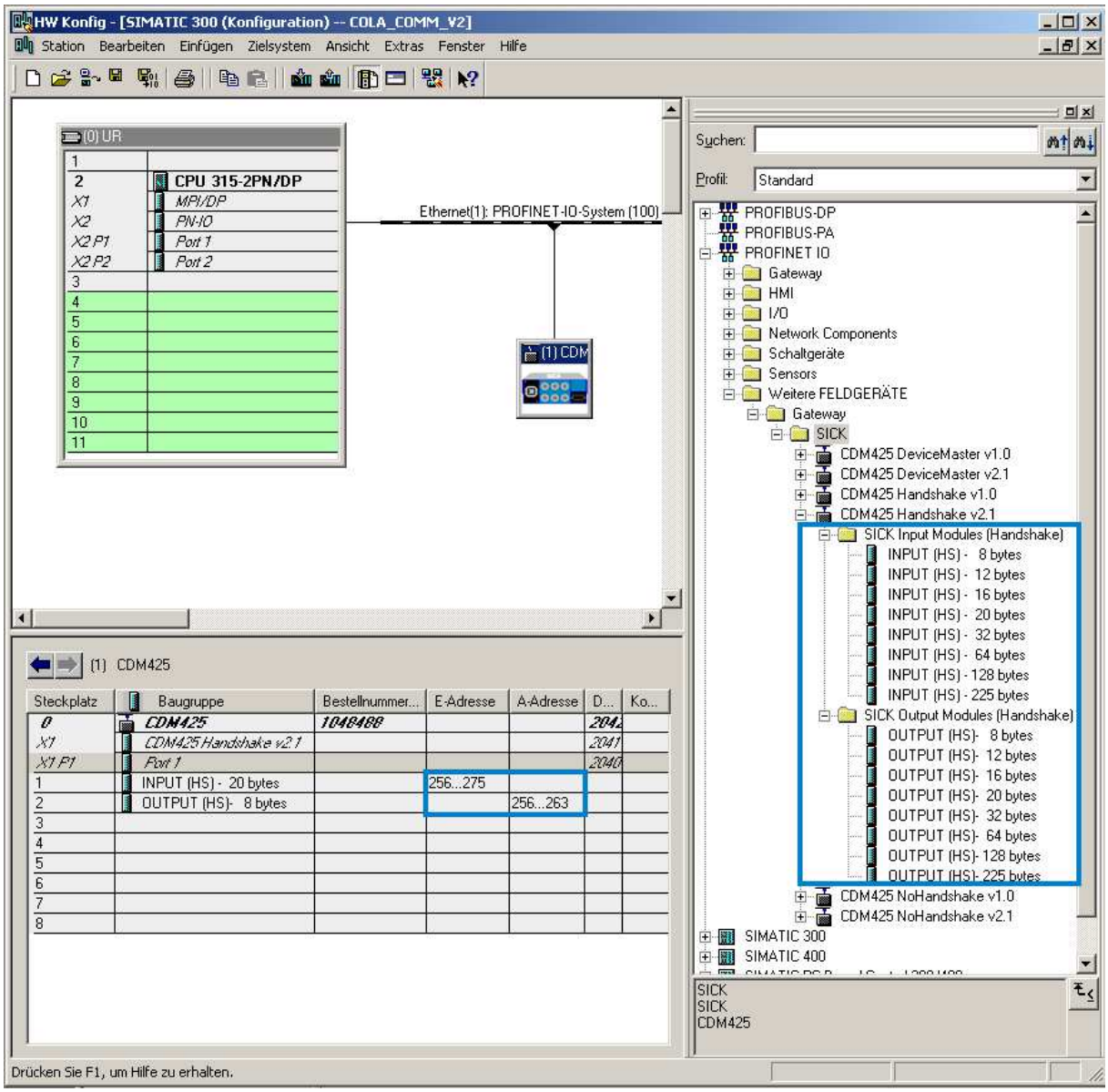

Figure 2: Step7 hardware configuration

4 Module Description

The CCOM_PNDP (FB10) function module simplifies the use of SICK sensors in combination with S7 PLC controllers. The module allows transmitting and receiving of CoLa telegrams by a Profibus/Profinet connection, which is projected in the hardware configuration.

The module supports following tasks:
- Sending of CoLa commands to a SICK sensor
- Receiving of CoLa answers from a SICK sensor
- Receiving of device transmitted telegrams (configurable in the SOPAS output format)

If the data cannot be transmitted /received in one cycle, the module will fragment them automatically.

The function block is working asynchronously, which means the processing is done via various function block call ups. Therefore it is necessary that the function block is called up cyclically in the user program.

The CCOM_PNDP (FB10) module encapsulates the Siemens function modules DPRD_DAT (SFC14) and DPWR_DAT (SFC15), which are used for the communication between SPS and Sensor (consistent data exchange).

4.1 Module Specification

Module Number:	FB10
Module Name:	CCOM_PNDP
Version:	2.1
Modules called:	SFC14 (DPRD_DAT) SFC15 (DPWR_DAT) SFC20 (BLKMOV) SFB4 (TON)
Data modules used:	-
Module call:	cyclic
Markers used:	none
Counter used:	none
Registers used:	AR1, AR2 (for multi instances)
Language used for module creation:	Step7-AWL

The system functions (SFCs) used in the function module must exist on the controller being used.

4.2 Operating principle

The following parameters must be specified before the CCOM_PNDP module can be used.

IN_ADDR: Projected entry point address of the used input modules of the Input-area. The entry point address is fixed by the projecting of the hardware (see chapter 3.3). The value has to be in hexadecimal format (e.g. address 256 = W#16#100).

IN_LEN: Length of the used input module in the hardware configuration. The length of the input module is fixed by the projecting of the hardware (see chapter 3.3).

OUT_ADDR: Projected entry point address of the used output module of the Output-area. The output address is fixed by the projecting of the hardware (see chapter 3.3). The value has to be in hexadecimal format (e.g. address 256 = W#16#100).

OUT_LEN: Length of the used output module in the hardware configuration. The length of the output module is fixed by the projecting of the hardware (see chapter 3.3).

COMMAND: The pointer shows the data range, where the CoLa command is archived. The data range has to be established by the programmer (e.g. data module with an Array of CHAR). The command must be specified without [STX] / [ETX] framing.

COMMAND_LEN: Byte length of the CoLa command.

RECORD: The pointer shows the data range, where the telegrams sent by the device, are achieved. The data range has to be established by the programmer (e.g. data module with an Array of Byte).

4.2.1 Receiving of Read Results (RD)

Data, sent by the device, are written in the RECORD, when the function module has received new data. The Bit RD_DONE shows the receiving of new data for one SPS cycle. As soon as the new data was received, the RD_COUNT counter is incremented. The particular byte length of the last received telegram can be taken out of the parameter RD_LEN.

4.2.2 Device Communication via CoLa Commands (REQ)

During the communication via CoLa commands, the order defined in COMMAND is transmitted to the device. The resulting answer is archived in the range defined by the pointer RECORD.

You start the transmission by triggering the START_REQ parameter with a rising edge.
As long as no valid answer to the transmitted CoLa command has arrived, this is signalized by the REQ_BUSY parameter. If no answer arrives during the Timeout (TOUT), the processing will be stopped with a time out error (REQ_ERRORCODE). The REQ_DONE output parameter shows, that an answer to a CoLa command was received (REQ_DONE = TRUE).

4.2.3 Timing

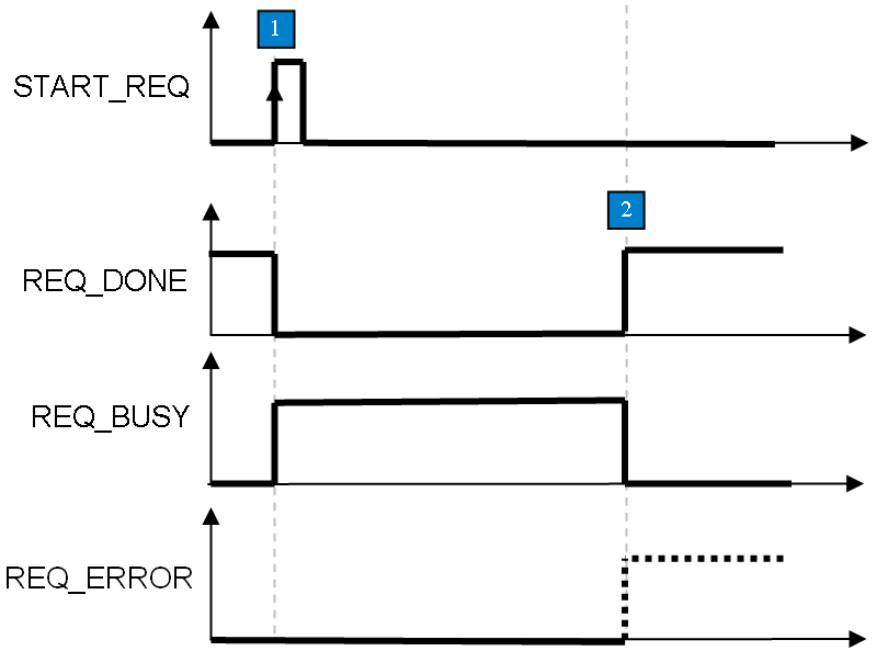

Figure 3: Timing Diagram

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

2: When all commands have been sent and all responses 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 terminated with “REQ_ERROR”.

4.3 Response to Errors

In case of an error the REQ_ERROR or RD_ERROR error bits signalize, that an error has appeared. In this case the REQ_ERRORCODE respectively RD_ERRORCODE parameter displays an error code. The REQ_ERROR error bit is set as long as no new order is started. If the error doesn’t exist anymore, the RD_ERROR parameter will only be active for one PLC-cycle and will be reset.

4.4 Reset of Communication

The RESET bit allows resetting the communication between gateway / sensor and PLC. In this connection the first 8 bytes of the peripheral output side are initialized with zero for one second. As soon as RESET = TRUE and START_REQ is triggered with a rising edge, the reset command is executed. The “REQ_BUSY” bit signalizes, that the order is edited. If the reset routine is completed, the “REQ_DONE” bit will be set.

4.5 Parameter

Parameter	Declaration	Data Type	Memory Area	Description
EN	INPUT	BOOL	I,M,D,L, const.	Enable (LD und FBD).
IN_ADDR	INPUT	WORD	I,M,D,L, const.	Projected starting address of the Input-area of the chosen module.
IN_LEN	INPUT	INT	I,M,D,L, const.	Length of the used input module in the hardware configuration. Valid value area: [8..128]
OUT_ADDR	INPUT	WORD	I,M,D,L, const.	Projected starting address of the Out-put-area of the chosen module.
OUT_LEN	INPUT	INT	I,M,D,L, const.	Length of the used output module in the hardware configuration. Valid value area: [8..128]
TOUT	INPUT	TIME	I,M,D,L, const.	Time after which a timeout error is provoked. If this parameter is not wired, the time out will be 5 seconds by default. Please note, that some CoLa commands need a longer process time (e.g. save commands)
START_REQ	INPUT	BOOL	I,M,D,L	Rising edge: Transmit a CoLa command and waits for the corresponding answers
RESET	INPUT	BOOL	I,M,D,L, const.	Reset of Communication (HS-Counter of the data protocol)
COMMAND	INPUT	ANY	D	Pointer on the range, which contains the transmitted CoLa command. Only the BYTE data type is allowed. The command must be specified without [STX] / [ETX] framing. Tip: Please note, that the parameter always needs the complete specification of the DB-parameters (e.g..: P#DB13.DBX0.0 BYTE 100). The omission of an explicit DB no. is forbidden and leads to a module error.
COMMAND_LEN	INPUT	INT	I,M,D,L, const.	Quantity of bytes, where the pointer references #COMMAND

Parameter	Declaration	Data Type	Memory Area	Description
RECORD	INPUT	ANY	D	Pointer on the range, where the device-sent telegrams should be archived. Only the data type BYTE is allowed. Tip: Please note, that the parameter always needs the complete specification of the DB-parameters (e.g..: P#DB13.DBX0.0 BYTE 100). The omission of an explicit DB no. is forbidden and leads to a module error.
RD_DONE	OUTPUT	BOOL	Q,M,D,L	Rising edge: A read result sent by the device was received (in SOPAS output format configurable) If a read result was received, the bit will be set for one PLC-cycle. The command answer is available in the memory area, which is referenced by the #Record parameter.
RD_ COUNT	OUTPUT	BYTE	Q,M,D,L	The parameter counts the quantity of the received read results. The counter counts from 0 to 255 (decimal). If there is an overrun, the counter will restart at zero.
RD_ LEN	OUTPUT	INT	Q,M,D,L	Byte length of the received read result
REQ_DONE	OUTPUT	BOOL	Q,M,D,L	Indicates whether a CoLa command was sent and an answer was received TRUE: editing finished FALSE: editing not yet finished The command answer is available in the memory area, which is referenced by the #Record parameter,
REQ_BUSY	OUTPUT	BOOL	Q,M,D,L	REQ charge is in editing
REQ_ LEN	OUTPUT	INT	Q,M,D,L	Length of an answer telegram in byte
REQ_ ERROR	OUTPUT	BOOL	Q,M,D,L	REQ error status: 0: No error 1: Abort with error
RD_ERROR	OUTPUT	BOOL	Q,M,D,L	RD error status: 0: No error 1: Abort 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 und FBD).

4.6 Error Codes

The REQ_ERRORCODE and RD_ERRORCODE parameters contain the following information.

Error Code	Brief description	Description
W#16#0000	No error	No error
W#16#0001	Invalid memory range of the #RECORD Pointer specified	Invalid memory range of the stated ANY-pointers. The pointer must have a dedicated DB.
W#16#0002	Invalid #RECORD pointer length specified	The length of the referenced data module is shorter than the length defined by the pointer.
W#16#0003	Invalid memory range of the #COMMAND Pointer specified	Invalid memory range of the stated ANY-pointer. The pointer must have a dedicated DB.
W#16#0004	Invalid #COMMAND pointer length specified	The length of the referenced data module 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 SPS - Communication parameter is incorrect - Application of CoLa commands, which doesn't send an answer (echo) - Editing time of the command > Timeout
W#16#0006	Invalid command length	The length of the transmitted command is longer than the specified command length (COMMAND_LEN).
W#16#0007	SFC20 error	The module SFC20 (BLKMOV) reports a module error. The error code is displayed in the "nStatusBLKMOV" variable of the instanced data module. . Please use the Step7 help system for an interpretation of the error codes
W#16#000A	Received telegram > length of #RECORD	The received telegram is longer than the specified length of #Record
W#16#000B	Invalid input module	The specified length of the input module is invalid (IN_LEN) Valid range of values: [8..128]
W#16#000C	Invalid output module	The specified length of the output module is invalid (OUT_LEN) Valid range of values: [8..128]
W#16#000D	Internal module error	Internal module error
W#16#8XXX	SFC14 / SFC15 error	The module SFC14 (DPRD_DAT) / SFC15 (DPWR_DAT) reports a module error. Please use the Step7 help system for an interpretation of the error codes

5 Example

Figure 4 shows a wiring example of the CCOM_PNDP FBs. A SICK device is projected with a data width of 20 bytes input and 8 bytes output in the hardware configuration. The selected I/O-modules are addressed to the starting address 256 (W#16#100) (see *Figure 2*).

Program call:

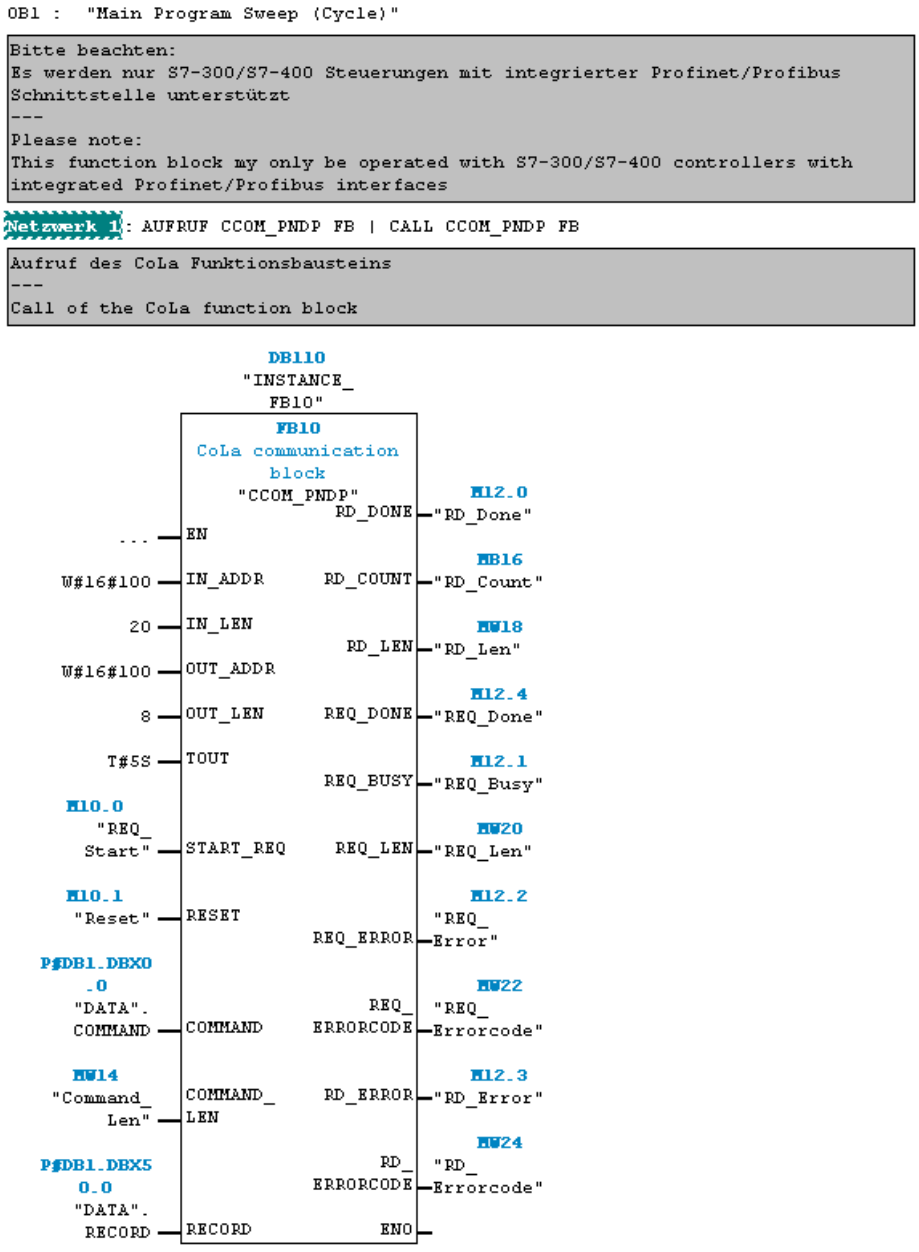

Figure 4: Call of CCOM_PNDP FBs in OB1

Table of variables for executing a CoLa command:

// CCOM PNDP integrated 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	

As soon as the “REQ_START” bit is accessed with a rising edge, the CoLa command (here: ’sMN mTCgateon’) is executed. The length of the order is transferred to the COMMAND_LEN parameter (here: 13 signs).

Table of variables for incoming command answers:

// 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

As soon as the value of the “REQ_DONE“ output bit changes from FALSE to TRUE (rising edge), the answer (REQ) to a transmitted command is available in the record range. The “REQ_LEN” parameter shows, how many bytes were received respectively are valid.

Table of variables for incoming read 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

As soon as a function module has received new data, data transmitted by a device is written in the record. The “RD_DONE” bit shows the receiving of new data for one SPS cycle (change of the signal from FALSE to TRUE). As soon as new data were received, RD_COUNT is incremented. The “RD_LEN” parameter shows, how many bytes were received respectively are valid.