Technical Information

SICK
CoLa Communication Block

CCOM_TCP Function Block for
Siemens S7 Controllers

Date of issue: March 15, 2012

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_TCP 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_TCP function block facilitates data exchange between SICK devices and S7 controllers. Only S7-300/S7-400 controllers with an integrated Industrial Ethernet (IE) interface are supported.

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

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

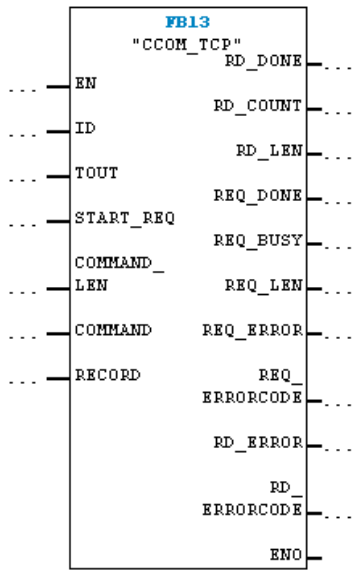

Figure 1: Representation of function block in FBD view

Optional functions:
- Send CoLa[1] commands to a SICK sensor
- Receive CoLa responses from a SICK sensor
- Receive telegrams sent by devices (can be configured in SOPAS[2] output format)

[1] The command language (CoLa) is a protocol internal to SICK for communicating with SOPAS devices.
[2] 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 and S7-400 controllers. Only controllers with an integrated IE interface are supported. TCP connections that have been configured using a communication processor (CP module) are not supported.

3.2 Establishing a connection

A TCP connection to the sensor must be established before the function block can be used. For S7 controllers with an integrated IE interface, Siemens provides the following communication blocks (Standard Library → Communication Blocks):

- FB65 (TCON): To establish a TCP connection
- FB65 (TDISCON): To close down a TCP connection
- FB63 (TSEND): To send data
- FB64 (TRCV): To receive data

Figure 2 shows the FB65 call with the associated instance (DB65) in OB1. When the controller starts up, OB100 is executed once. The start bit (REQ) of FB65 is set in OB100 to establish the connection via FB65. Successful connection setup is indicated by the bit DONE = TRUE. The connection parameters for establishing a connection are saved in a data structure (UDT65).

The ID parameter of the TCON block and the instantiated UDT structure must be identical to the ID of the CCOM_TCP function block.

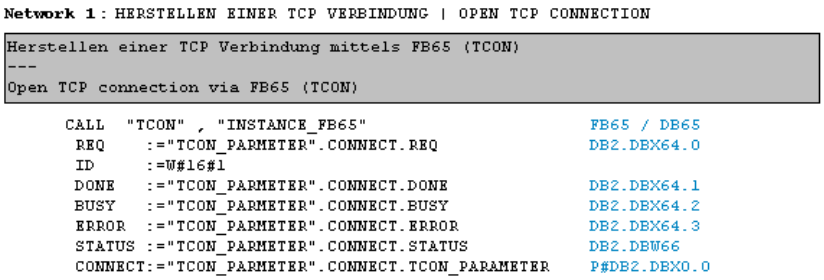

Figure 2: Using FB65 (TCON) to establish a TCP connection

The table below shows an example configuration of the UDT65.

Byte	Parameter	Data type	Start value	Description
0 - 1	block_length	WORD	W#16#0040	Length of the UDT65: 64 bytes (fixed)
2 - 3	id	WORD	W#16#0001	Reference to TCP connection. This value must be identical to the ID parameter at TCON (FB65) and to the CCOM_TCP (FB13) block.
4	connection_type	BYTE	B#16#11	Connection type = TCP
5	active_est	BOOL	TRUE	Active connection establishment

Byte	Parameter	Data type	Start value	Description
6	local_device_id	BYTE	B#16#02	Type of TCP connection In this case: Communication via the integrated Ethernet interface for CPUs 315-2 PN/DP and 317-2 PN/DP
7	local_tsap_id_len	BYTE	B#16#02	For connection type B#16#11 and a passive end point
8	rem_subnet_id_ len	BYTE	B#16#00	Not used
9	rem_staddr_len	BYTE	B#16#04	Length of the IP address of the station (SICK device)
10	rem_tsap_id_len	BYTE	B#16#02	Fixed for connection type B#16#11
11	Next_staddr_len	BYTE	B#16#00	Used length of the next_staddr parameter (not used)
12 - 27	Local_tsap_id	Array [1..16] of BYTE	-	Number of the local port used: [1] = High byte of the port number used represented in hex format [2] = Low byte of the port number used represented in hex format [3..16] = B#16#00
28 - 33	rem_subnet_id	Array [1..6] of BYTE	-	Not used [1..16] = B#16#0
34 - 39	Rem_staddr	Array [1..6] of BYTE	-	IP address of SICK device For example: 192.168.10.15 [1] = B#16#C0 (192) [2] = B#16#A8 (168) [3] = B#16#0A (10) [4] = B#16#0F (15) [5-6] = B#16#00
40 - 55	Rem_tsap_id	Array [1..16] of BYTE	-	Port number of connected SICK device SICK communication port: 2112 (decimal) [1] = B#16#08 (high byte) [2] = B#16#40 (low byte) [3..16] = B#16#0
56 - 61	Next_staddr	Array [1..6] of BYTE	-	Not used [1..6] = B#16#0
62 - 63	spare	WORD	W#16#0000	Not used

Please refer to the Step7 help system for a precise description of the UDT parameters.

The CCOM_TCP block will only function if there is an active TCP connection to the SICK device. Figure 3 shows the Step7 diagnostics screen for open communication via Industrial Ethernet. To access this screen, select "Module Status → Diagnostics → Occupied Connection Resources".

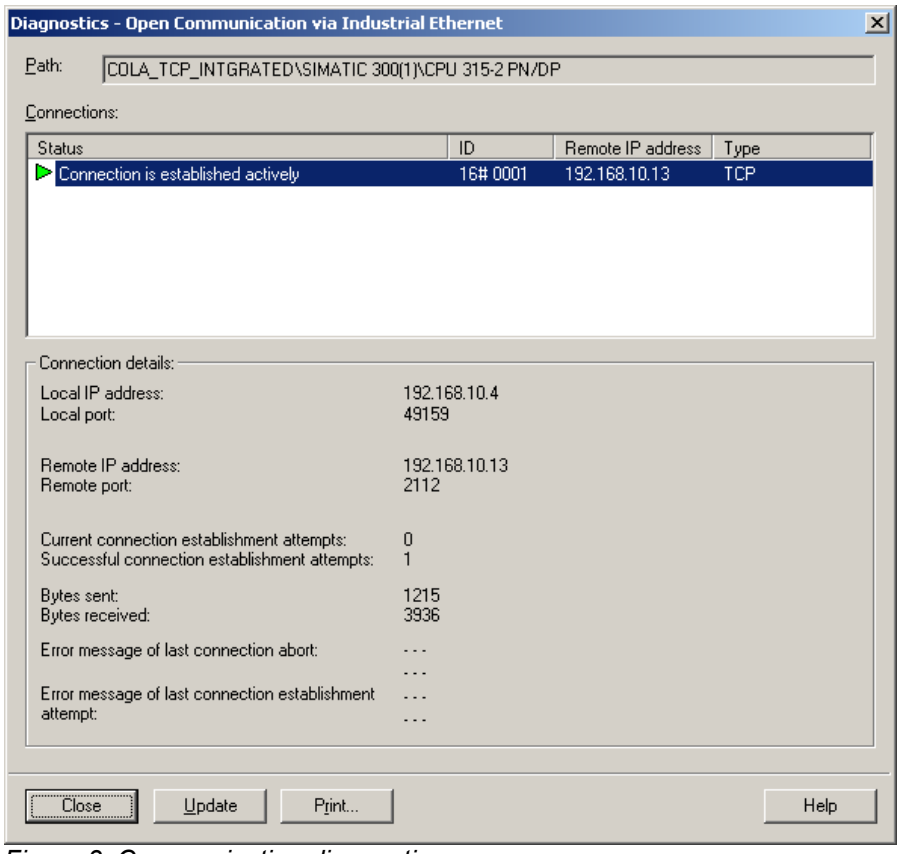

Figure 3: Communication diagnostics

4 Block description

The CCOM_TCP (FB13) function block makes it easier to use SICK sensors with S7 controllers. The block enables you to send and receive CoLa telegrams via a TCP connection that has been configured with FB65.

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 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.

The CCOM_TCP block (FB13) encapsulates Siemens function blocks TSEND (FB63) and TRCV (FB64), which are used for communication between the PLC and sensor.

4.1 Block specifications

Block number:	FB13
Block name:	CCOM_TCP
Version:	1.0
Blocks called:	FB63 (TSEND) FB64 (TRCV) 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

Blocks FB63 and FB64 are provided by the Siemens library.

4.2 Operating principle

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

ID: Connection ID of the TCP connection. The value specified here must be the same as for the ID parameter of the TCON block and the instantiated UDT structure. See also Figure 3.

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 commands must always be specified with [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. CoLa commands are always sent with control characters ([ETX], [STX] framing).

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

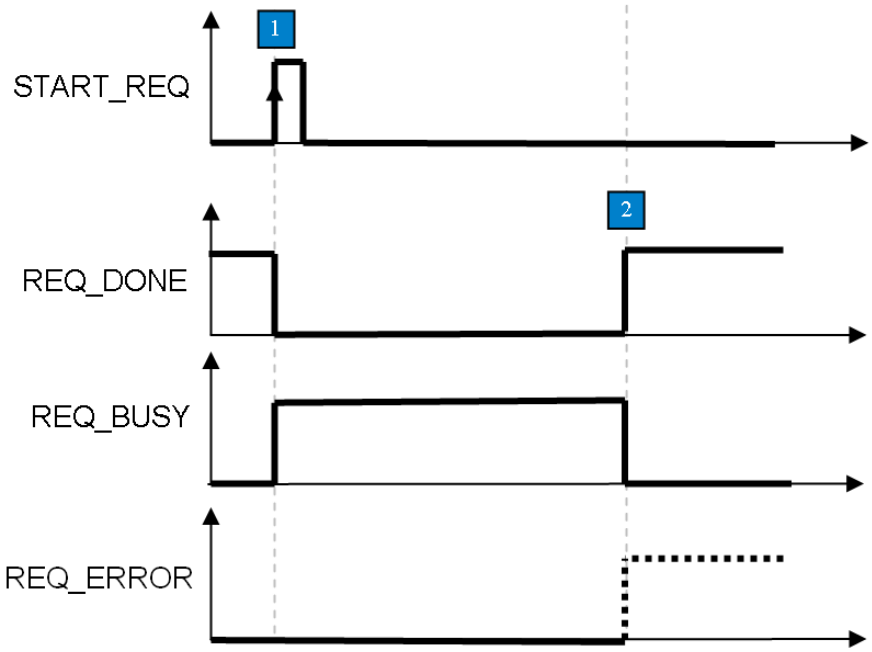

Figure 4: Timing diagram

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 Parameters

Parameter	Declaration	Data type	Memory area	Description
EN	INPUT	BOOL	I,M,D,L, const.	Enable input (LD and FBD)
ID	INPUT	INT	I,M,D,L, const.	Connection ID for configured TCP connection (see communication diagnostics Figure 3 or ID parameter TCON FB)
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
COMMAND	INPUT	ANY	D	Pointer referencing the area containing the CoLa command to be sent. Only the BYTE data type is permitted. The commands must always be specified with [STX]/[ETX] framing (ASCII control characters). 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.

Parameter	Declaration	Data type	Memory area	Description
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
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_ERROR CODE	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)

4.5 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 selected 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	Invalid CoLa command	There is no [STX] [ETX] framing for the specified CoLa command.
W#16#000A	Telegram received > #RECORD length	The received telegram is longer than the specified #RECORD length.
W#16#000B	Telegram received without control characters	- A telegram was received without [STX] [ETX] framing. - The received telegram is longer than the specified #RECORD length.
W#16#7XXX - W#16#8XXX	FB63/FB64 error	For a description of the error, see the Step7 help system.

5 Example

Figure 7 shows an example of a connected CCOM_TCP function block. The TCP connection to the SICK sensor is established with FB65 (TCON) during PLC startup (see Figure 5 / Figure 6).

Program call:

```
OB100 :  "Complete Restart"
Comment:

Network 1: TCP VERBINDUNG HERSTELLEN | ESTABLISHing A TCP CONNECTION
Herstellen einer TCP Verbindung nach jedem SPS restart.
---
Open TCP connection after every PLC restart.

      SET
      =     "TCON_PARMETER".CONNECT.REQ    DB2.DBX64.0
```

Figure 5: Start of the connection setup in OB100

```
OB1 :  "Main Program Sweep (Cycle)"
Bitte beachten:
Es werden nur S7-300/S7-400 Steuerungen mit integrierter TCP-Schnittstelle
unterstützt
---
Please note:
This function block my only be operated with S7-300/S7-400 controllers with
integrated TCP interfaces

Network 1: HERSTELLEN EINER TCP VERBINDUNG | OPEN TCP CONNECTION
Herstellen einer TCP Verbindung mittels FB65 (TCON)
---
Open TCP connection via FB65 (TCON)

      CALL  "TCON" , "INSTANCE_FB65"                    FB65 / DB65
       REQ     :="TCON_PARMETER".CONNECT.REQ            DB2.DBX64.0
       ID      :=W#16#1
       DONE    :="TCON_PARMETER".CONNECT.DONE           DB2.DBX64.1
       BUSY    :="TCON_PARMETER".CONNECT.BUSY           DB2.DBX64.2
       ERROR   :="TCON_PARMETER".CONNECT.ERROR          DB2.DBX64.3
       STATUS  :="TCON_PARMETER".CONNECT.STATUS         DB2.DBW66
       CONNECT:="TCON_PARMETER".CONNECT.TCON_PARAMETER  P#DB2.DBX0.0

      CLR
      =     "TCON_PARMETER".CONNECT.REQ                DB2.DBX64.0
```

Figure 6: FB65 (TCON) call for creating a TCP connection

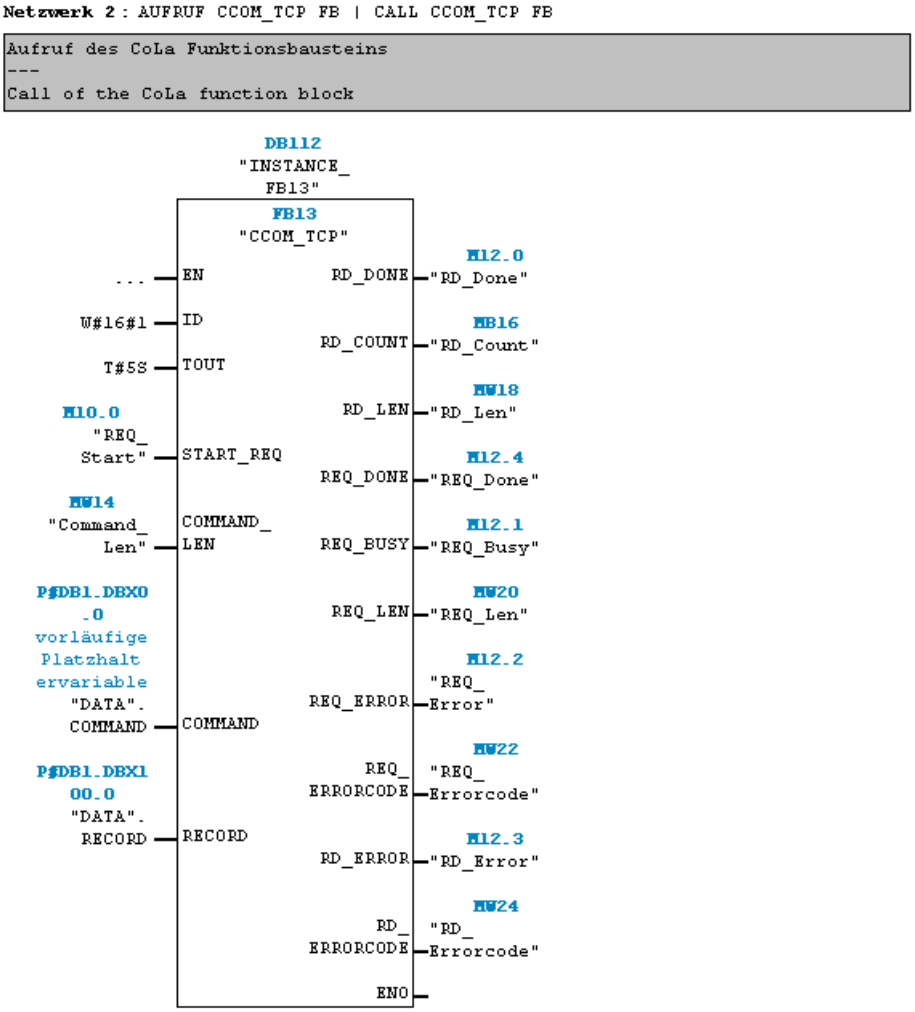

Figure 7: CCOM_TCP communication block call

Table of variables for executing a CoLa command:

// CCOM TCP Integrated function block				
M 10.0	"REQ_Start"	BOOL	true	
// Reading Result Status				
M 12.0	"RD_Done"	BOOL	false	
MB 16	"RD_Count"	DEC	23	
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"	DEC	15	15
DB1.DBB 0	"DATA".COMMAND[1]	CHARACTER	'['	'['
DB1.DBB 1	"DATA".COMMAND[2]	CHARACTER	's'	's'
DB1.DBB 2	"DATA".COMMAND[3]	CHARACTER	'M'	'M'
DB1.DBB 3	"DATA".COMMAND[4]	CHARACTER	'N'	'N'
DB1.DBB 4	"DATA".COMMAND[5]	CHARACTER	' '	' '
DB1.DBB 5	"DATA".COMMAND[6]	CHARACTER	'm'	'm'
DB1.DBB 6	"DATA".COMMAND[7]	CHARACTER	'T'	'T'
DB1.DBB 7	"DATA".COMMAND[8]	CHARACTER	'C'	'C'
DB1.DBB 8	"DATA".COMMAND[9]	CHARACTER	'g'	'g'
DB1.DBB 9	"DATA".COMMAND[10]	CHARACTER	'a'	'a'
DB1.DBB 10	"DATA".COMMAND[11]	CHARACTER	't'	't'
DB1.DBB 11	"DATA".COMMAND[12]	CHARACTER	'e'	'e'
DB1.DBB 12	"DATA".COMMAND[13]	CHARACTER	'o'	'o'
DB1.DBB 13	"DATA".COMMAND[14]	CHARACTER	'n'	'n'
DB1.DBB 14	"DATA".COMMAND[15]	CHARACTER	']'	']'
DB1.DBB 15	"DATA".COMMAND[16]	CHARACTER	B#16#00	
DB1.DBB 16	"DATA".COMMAND[17]	CHARACTER	B#16#00	
DB1.DBB 17	"DATA".COMMAND[18]	CHARACTER	B#16#00	
DB1.DBB 18	"DATA".COMMAND[19]	CHARACTER	B#16#00	
DB1.DBB 19	"DATA".COMMAND[20]	CHARACTER	B#16#00	

The CoLa command ("[STX]sMN mTCgateon[ETX]" 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: 15 characters).

Table of variables for incoming command responses:

// Record			
MW 18	"RD_Len"	DEC	50
MW 20	"REQ_Len"	DEC	17
DB1.DBB 100	"DATA".RECORD[1]	CHARACTER	'┐'
DB1.DBB 101	"DATA".RECORD[2]	CHARACTER	's'
DB1.DBB 102	"DATA".RECORD[3]	CHARACTER	'A'
DB1.DBB 103	"DATA".RECORD[4]	CHARACTER	'N'
DB1.DBB 104	"DATA".RECORD[5]	CHARACTER	' '
DB1.DBB 105	"DATA".RECORD[6]	CHARACTER	'm'
DB1.DBB 106	"DATA".RECORD[7]	CHARACTER	'T'
DB1.DBB 107	"DATA".RECORD[8]	CHARACTER	'C'
DB1.DBB 108	"DATA".RECORD[9]	CHARACTER	'g'
DB1.DBB 109	"DATA".RECORD[10]	CHARACTER	'a'
DB1.DBB 110	"DATA".RECORD[11]	CHARACTER	't'
DB1.DBB 111	"DATA".RECORD[12]	CHARACTER	'e'
DB1.DBB 112	"DATA".RECORD[13]	CHARACTER	'o'
DB1.DBB 113	"DATA".RECORD[14]	CHARACTER	'n'
DB1.DBB 114	"DATA".RECORD[15]	CHARACTER	' '
DB1.DBB 115	"DATA".RECORD[16]	CHARACTER	'1'
DB1.DBB 116	"DATA".RECORD[17]	CHARACTER	'└'
DB1.DBB 117	"DATA".RECORD[18]	CHARACTER	B#16#00
DB1.DBB 118	"DATA".RECORD[19]	CHARACTER	B#16#00
DB1.DBB 119	"DATA".RECORD[20]	CHARACTER	B#16#00

The response (REQ) to a sent command (in this case: "[STX]sAN mTCgateon 1[ETX]") 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"	DEC	11
MW 20	"REQ_Len"	DEC	17
DB1.DBB 100	"DATA".RECORD[1]	CHARACTER	'ך'
DB1.DBB 101	"DATA".RECORD[2]	CHARACTER	'1'
DB1.DBB 102	"DATA".RECORD[3]	CHARACTER	'2'
DB1.DBB 103	"DATA".RECORD[4]	CHARACTER	'3'
DB1.DBB 104	"DATA".RECORD[5]	CHARACTER	'4'
DB1.DBB 105	"DATA".RECORD[6]	CHARACTER	'5'
DB1.DBB 106	"DATA".RECORD[7]	CHARACTER	'6'
DB1.DBB 107	"DATA".RECORD[8]	CHARACTER	'7'
DB1.DBB 108	"DATA".RECORD[9]	CHARACTER	'8'
DB1.DBB 109	"DATA".RECORD[10]	CHARACTER	'9'
DB1.DBB 110	"DATA".RECORD[11]	CHARACTER	'ט'
DB1.DBB 111	"DATA".RECORD[12]	CHARACTER	'e'
DB1.DBB 112	"DATA".RECORD[13]	CHARACTER	'o'
DB1.DBB 113	"DATA".RECORD[14]	CHARACTER	'n'
DB1.DBB 114	"DATA".RECORD[15]	CHARACTER	' '
DB1.DBB 115	"DATA".RECORD[16]	CHARACTER	'1'
DB1.DBB 116	"DATA".RECORD[17]	CHARACTER	'ט'
DB1.DBB 117	"DATA".RECORD[18]	CHARACTER	B#16#00
DB1.DBB 118	"DATA".RECORD[19]	CHARACTER	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.