Technical Information

# SICK
# LECTOR / CLV6xx Function Block

**Version V2.X**

SICK LECTOR CLV TCP CP Function Block for
Siemens Step7 Controls


SICK
Sensor Intelligence.

# Table of Content

# 1 About this document

Please read this chapter carefully before you start working with this Technical Information and with SICK Lector/CLV6xx function block.

## 1.1 Function of this document

This Technical Information describes how to use SICK LECTOR CLV6XX TCP CP function block. It is used for guiding technical personnel working for the machine manufacturer / operator in project planning and commissioning.

## 1.2 Target group

This Technical Information is aimed for specialists, such as technicians and engineers.

# 2 General Information

The function block „SICK LECTOR CLV TCP CP“ is used for the communication between a SIMATIC control and a SICK Lector 2D code reader or a CLV6xx barcode reader. The code reader communicates with the control via a TCP connection.

The following image shows the function block in the view of the function plan (FUP).

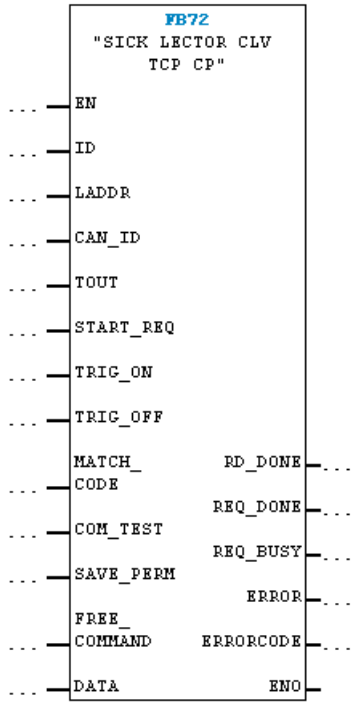



*Image 1: SICK LECTOR CLV TCP CP function block*

Features of function block:

- Sending of a trigger (CoLa[i] command) via the PLC
- Receiving of read results (defined in SOPAS-ET[ii] output format)
- Create / change an evaluation condition for a matchcode
- Carrying out a communication test
- Permanent saving of all device parameters in the device
- Communication via free selectable CoLa commands (CoLa-A protocol)
- Addressing of devices which communicate via CAN-Bus

[i] The Command Language (CoLa) is an internal SICK protocol for the communication with SOPAS devices
[ii] SOPAS-ET is an engineering tool for the configuration of SICK sensors

# 3 Hardware Configuration

## 3.1 Supported PLC controls

The function block must only be used with a Simatic S7 control family 300. Only controls which use a CP module for the TCP communication are being supported. Controls with integrated Ethernet interface are not supported,

## 3.2 Configuration in Step7

Before the function block can be used, a TCP connection to the sensor has to be set. Therefore you have to open the NetPro connection tool in the Simatic Manager (Simatic Manger → Extras → configure network).

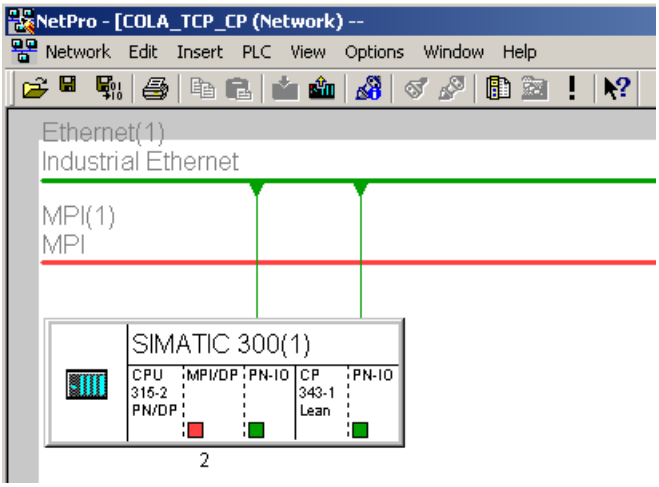



*Image 2: NetPro (Step7)*

Please mark the CPU in the S7-Station and insert a new connection at „Insert → New Connection“ as an unspecified TCP-connection.

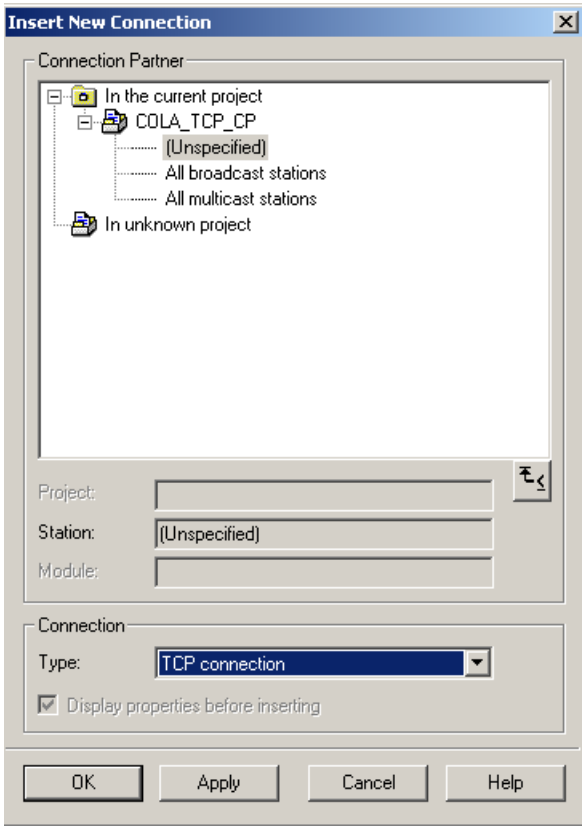



*Image 3: Insert a TCP-Connection in NetPro (Step7)*

In the dialogue of the TCP-Connection the checkbox „active connection setting“ has to be activated in the index „General“. The connection parameters for the TCP-connection are indicated on the right hand side of the dialogue. These values have to be transferred to the SICK LECTOR CLV TCP CP function block at the call-up (Parameter: ID / LADDR).

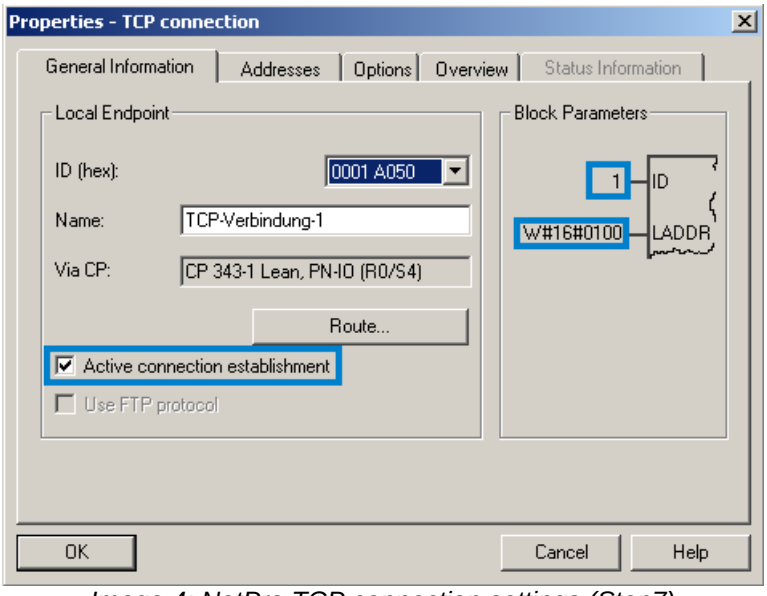



*Image 4: NetPro TCP connection settings (Step7)*

The IP-address and the used port of the SICK device have to be inserted at the index „Addresses“. SICK sensors generally use Port 2112 for the communication.

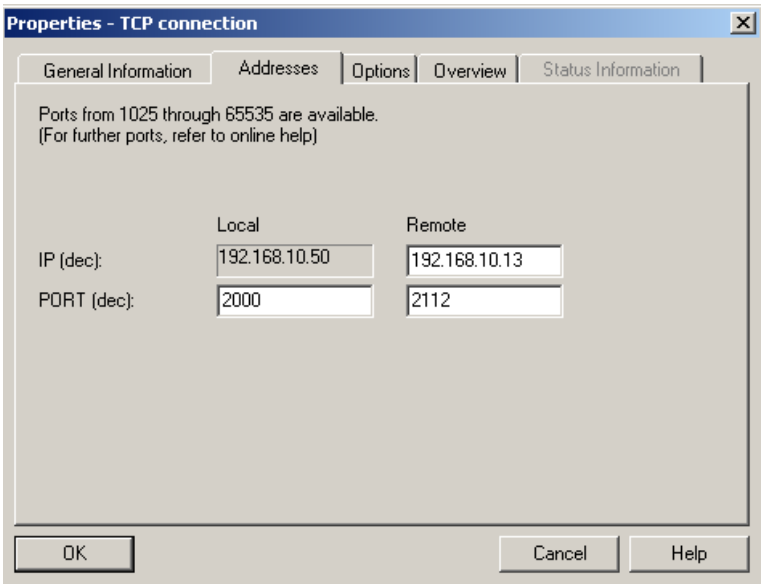



*Image 5: Addresses of the TCP-Connection (Step7)*

In the index „Options“ the mode „Send/Recv“ has to be set.

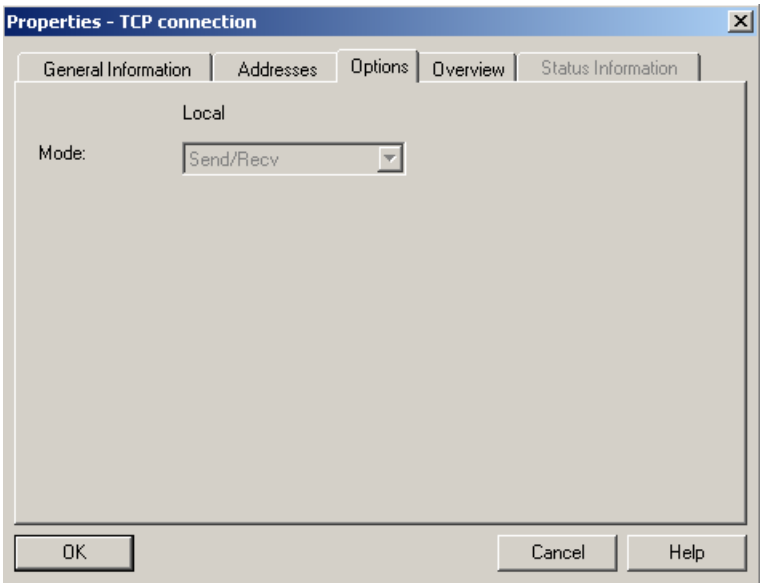



*Image 6: Mode of the TCP-Connection (Step7)*

After having closes the dialogue with „OK“, automatically a TCP-Connection is shown in the connection table. Now the station has to be saved and then the connection has to be loaded into the S7-Control. You might have to start the sensor once again in order to establish the TCP-Connection.

In order to diagnose the projected TCP-Connection, the connection status can be seen at „End system → Activate connection status“.

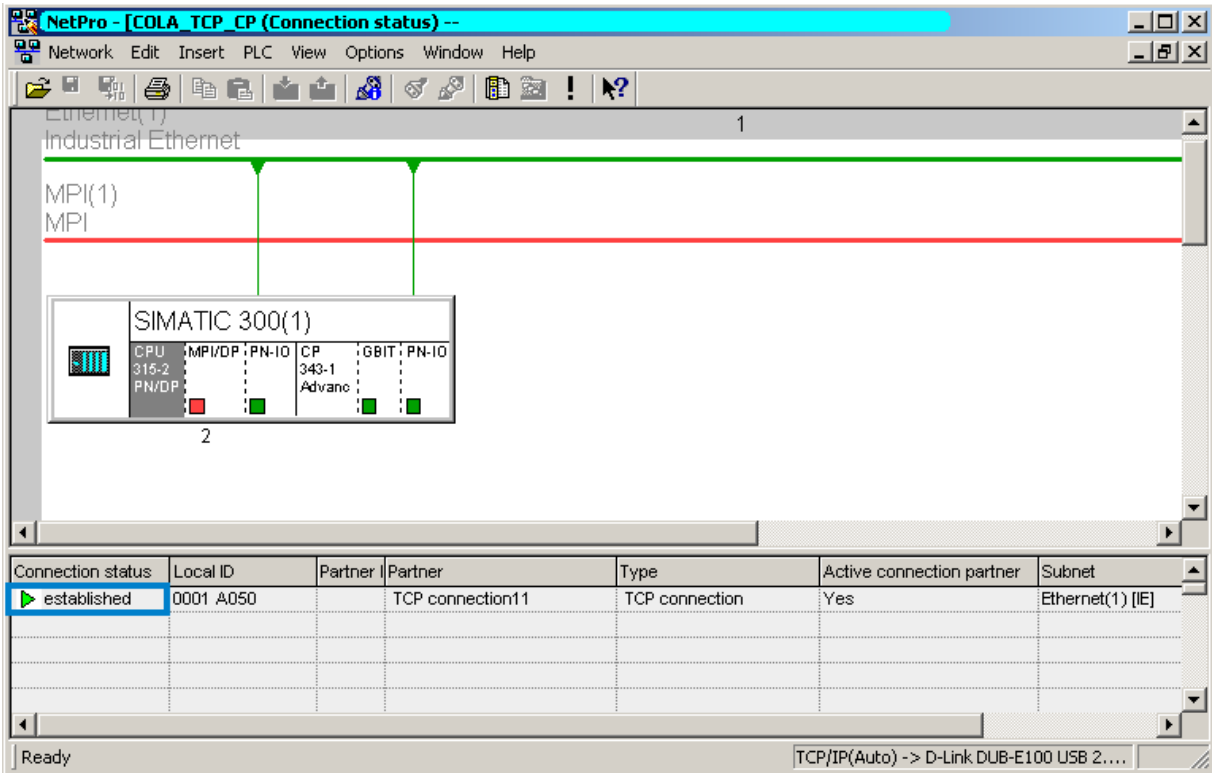



*Image 7: Connection status of the TCP-Connection (Step7)*

## 3.3 SOPAS device configuration

That a trigger via the PLC can take place, the trigger source has to be set before to „command“ with SOPAS-ET. ImageImage 8 shows how CLV6xx or Lector is configured at the menu point „Object Trigger“.

- Start with "SOPAS-Command" (TRIG_ON command has to be used)
- Stop with "SOPAS-Command" (TRIG_OFF command has to be used) Optionally the trigger window can automatically be closed, if the sensor has read a code with „Good Read“ or, in the case of a „No Reads“, after a defined timeout (here 500ms).

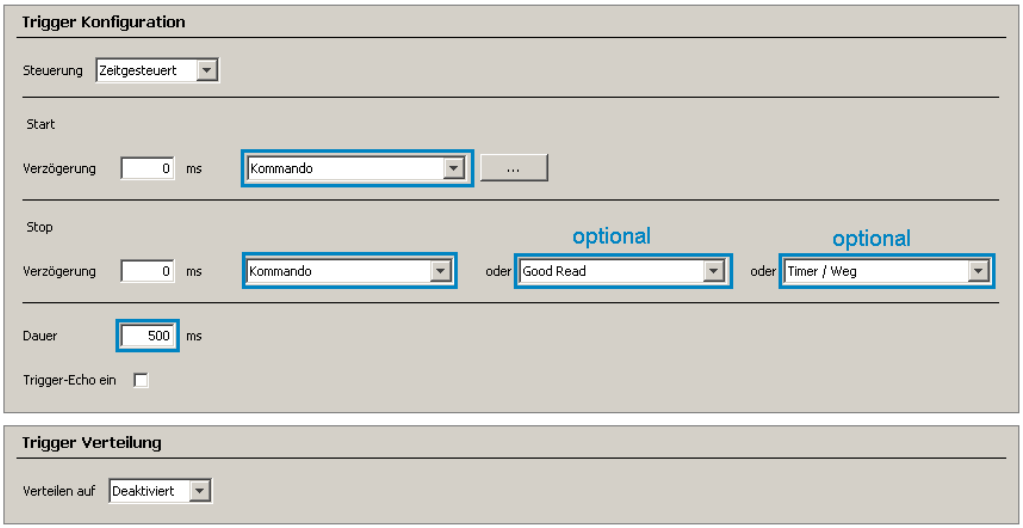



*Image 8: SOPAS Trigger settings*

If the device should be triggered directly, e.g. via a light switch or a hardware signal at the Sensor1-entry of LECTOR/CLV, the function blocks TRIG_ON / TRIG_OFF cannot be used anymore. If a trigger result has been received from the function block, this is signalized via the output parameter „RD_DONE“.

The read results always have to be differentiated to the SOPAS commands. If the read result can start with „s“ (lower case character „s“ as a SOPAS command), the output format has to be changed by placing e.g. a „D“ in front.

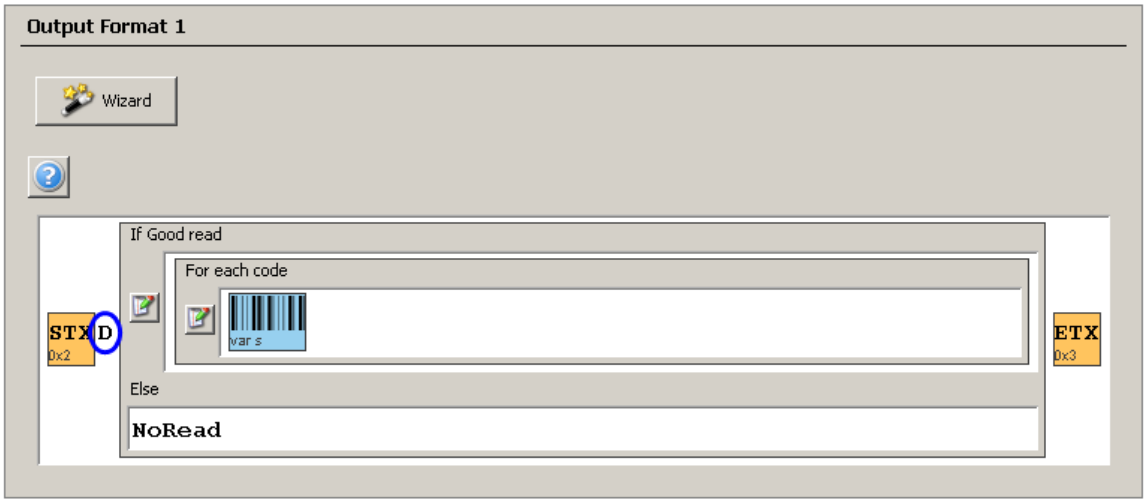



*Image 9: SOPAS output format settings*

# 4 Description of function block

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 function block encapsulates „SICK CCOM TCP CP“ (FB11), which allows the communication between PLC and sensor.

## 4.1 Function block specification

| | |
|---|---|
| Number of function block: | FB72 |
| Name of function block: | SICK LECTOR CLV TCP CP |
| Version: | 2.0 |
| Called-up function blocks: | FC5 (AG_SEND)<br>FC6 (AG_RECV)<br>SFC20 (BLKMOV)<br>SFB4 (TON)<br>FB11 (SICK CCOM TCP CP)<br>FB103 (AG_RECV_TCP_xVAR) |
| Used data blocks: | DB72 (SICK LECTOR CLV DATA) |
| Function block call up: | Cyclically |
| Used flag: | none |
| Used counter: | none |
| Used register: | AR1, AR2 (for multi instance call up) |
| Multi instant capable: | yes |
| Language: | Step7-AWL |
| Step7 Version: | Simatic Step7 V5.5 |

The system functions (SFCs) used in the function block have to be available on the respective PLC.

When changing the function block numbers, the respective calls in the function block SICK LECTOR CLV TCP CP have to be updated.

## 4.2 Operation Mode

In order to use the function block, the following communication parameters have to be set:

ID: Connection-ID of the TCP-Connection. The parameter is shown in the NetPro connectivity features (see Image 4).

LADDR: Starting address of the CP-component. The parameter is shown in the NetPro connectivity features (see Image 4).

DATA: The data block (DB72) belonging to the function block contains in- and output parameter of the supported function block actions. The data block has to be transferred to the input parameter „DATA“ of the function block.

Realizable function block functions:

- Trigger on → Opens the reading gate of the device per CoLa command
- Trigger off → Closes the reading gate of the device per CoLa command
- Match Code → Creates / changes a new evaluation condition for a matchcode
- Communication test → Checks if the device can be reached via „sRI0“ command
- Save Permanent → Saves all device parameters permanently in the device
- Free Command → Executes a free selectable CoLa command

In order to execute a function block action (TRIG_ON, TRIG_OFF, etc.), the desired action has to be selected first. Only one action can be executed at the same time. In order to do the action, the parameter START_REQ has to be triggered with a positive edge (signal change from a logical cero to one). As long as no valid device answer has to be received, this is signalized via the parameter REQ_BUSY

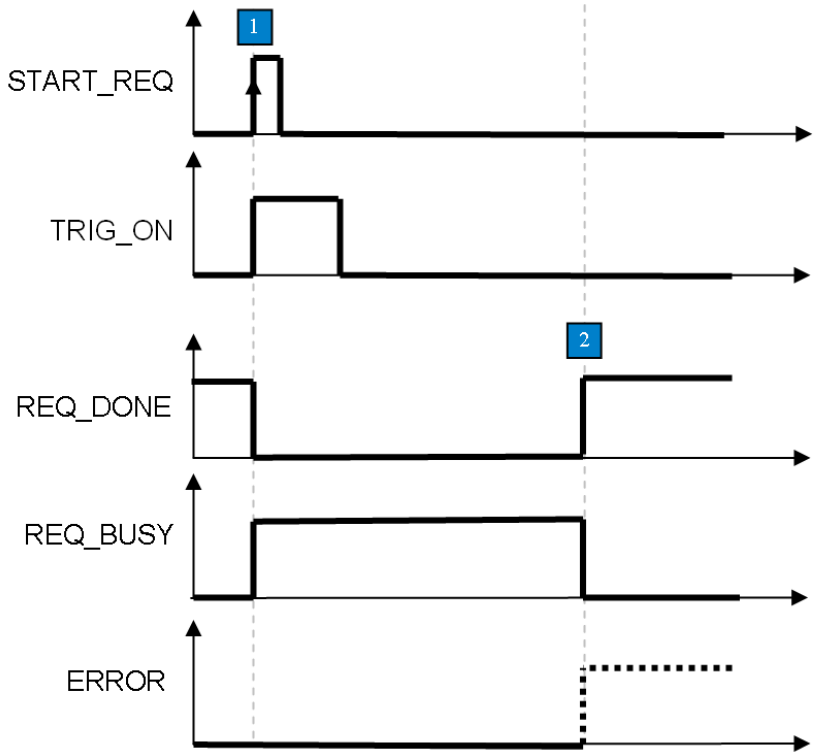
If the function block signalizes REQ_DONE = TRUE at the output parameter, the action has been done successfully. If, for this action (e.g. FREE_COMMAND) data has been requested from the device, it will be copied in the respective data area of the UDTs.

Data that is sent per trigger (TRIG_ON, TRIG_OFF) or directly from the device (e.g. direct trigger via a light switch), is stored in the data function block (ReadingResult.arrResult). The output parameter RD_DONE indicates for one PLC cycle, that new data has been received. The from the device sent data can be changed in the SOPAS output format (see chapter 3.3).

## 4.3 Behavior in the case of an error

If there is a wrong input value or a wrong input circuit of the function block, an error bit (*bError*) is set and an error code (*iErrorcode*) will be given out. In this case there is no further processing. The diagnosis parameter (*bError* and *iErrorcode*) of the routine maintain their value until a new request has been started.

## 4.4 Timing




1: Request via Pos edge to START_REQ
The desired action (here TRIG_ON) has to be selected in advance / at the same time. Only one action must be selected at the same time, otherwise there is a break-down with „ERROR“.

2: If all commands are sent and all replies are received, the action is ended with *bReqDone* = *TRUE*. If the action is faulty, it will be terminated with *bError* = *TRUE*. If terminated with *bError*, you can find the error in *iErrorcode*.

## 4.5 Value Transfer

The data function block „SICK LECTOR CLV DATA“ (DB72) contains input and output parameters of all supported function block actions. The data function block can be re-named according to the user program. The data structure is pre-defined and must not be changed (except for the last entry (ReadingResult.arrResult) (see chapter 4.6: Receipt of read results > 200 Byte).

DB72 -- "SICK LECTOR CLV DATA" -- SICK_LECTOR_CLV_PNDP_LIB\S7 Program(1)\...\DB72

| Address | Name | Type | Initial value | Comment |
|---|---|---|---|---|
| 0.0 | | STRUCT | | |
| +0.0 | Matchcode | STRUCT | | -- MATCH CODE -- |
| +0.0 | nMatchNumber | BYTE | B#16#1 | Matchcode number (Match1..9) (IN) |
| +1.0 | nCodeType | CHAR | ' ' | Codetype see device docu. (Example: 'd'= EAN-Code; '*'= Don't care) (IN) |
| +2.0 | nLength | BYTE | B#16#0 | Sets min and max length. B#16#0 = Don't care (IN) |
| +4.0 | iContentLength | INT | 0 | Content length (IN) |
| +6.0 | arrContent | ARRAY[1..75] | | Matchcode content (IN) |
| *1.0 | | CHAR | | |
| =82.0 | | END_STRUCT | | |
| +82.0 | FreeCommand | STRUCT | | -- FREE COMMAND -- |
| +0.0 | iCommandLength | INT | 0 | Length of the free command (IN) |
| +2.0 | arrCommand | ARRAY[1..100] | | Command (SICK CoLa-A protocol without [STX]/[ETX] framing) (IN) |
| *1.0 | | CHAR | | |
| +102.0 | iResultLength | INT | 0 | Byte length of the free command result (OUT) |
| +104.0 | arrResult | ARRAY[1..100] | | Result (SICK CoLA-A protocol) (OUT) |
| *1.0 | | CHAR | | |
| =204.0 | | END_STRUCT | | |
| +286.0 | ReadingResult | STRUCT | | -- READING RESULT -- |
| +0.0 | nCounter | BYTE | B#16#0 | This counter is incremented if a new reading result has arrived (OUT) |
| +2.0 | iLength | INT | 0 | Byte length of the reading result (OUT) |
| +4.0 | arrResult | ARRAY[1..200] | | Reading result data (OUT) |
| *1.0 | | CHAR | | |
| =204.0 | | END_STRUCT | | |
| =490.0 | | END_STRUCT | | |

*Image 10: Structure of the SICK LECTOR CLV DATA*

### 4.5.1 Matchcode

With the help of the matchcode action you have the possibility to create a new evaluation condition or to change an existing one. Before the matchcode action can be carried out, the following parameters have to be set in the structure *Matchcode*.

| Parameter | Declaration | Data Type | Description |
|---|---|---|---|
| Matchcode.<br>iMatchNumber | Input | BYTE | The matchcode number defines the name of the evaluation condition (e.g. 1=Match1, 2=Match2).<br><br>Valid value area:<br>[1..9] |
| Matchcode.<br>CodeType | Input | CHAR | Desired code type which the evaluation condition should refer to (e.g. 'w'= Datamatrix, 's'= QR-Code, '*' = any code type).<br><br>A selection of all code types can be found in the device documentation. |
| Matchcode.<br>nLength | Output | BYTE | Minimal and maximal code length<br>0 = any code length |
| Matchcode.<br>iContentLength | Output | INT | Byte length of the given matchcode |
| Matchcode.<br>arrContent | Output | ARRAY [1..75] OF CHAR | Matchcode content |

*Table 1: Matchcode Parameter*

### 4.5.2 Free Command

With the help of a free command you have the possibility to communicate via a valid CoLa command with the device. Hence it is necessary to store the command in the parameter *arrCommand* of the structure *FreeCommand*. The character length of the transferring command is written in the parameter „iCommandLength“.The commands can be looked up in the device description or SOPAS-ET.

| Parameter | Declaration | Data type | Description |
|---|---|---|---|
| FreeCommand. iCommandLength | Input | INT | Character length of the transferring CoLa command.<br><br>Valid value area:<br>[1..100] |
| FreeCommand. arrCommand | Input | ARRAY [1..100] OF CHAR | Free selectable CoLa command (Commands can be looked up in the device documentation). |
| FreeCommand. iResultLength | Output | INT | Byte length of the receiving CoLa telegram. |
| FreeCommand. arrResult | Output | ARRAY [1..100] OF CHAR | Receiving answer of the sent CoLa telegram. |

*Table 2: Free Command Parameter*

### 4.5.3 Reading Result

In the array „ReadingResult.arrResult“ data is stored, which is sent via trigger order (TRIG_ON, TRIG_OFF) or directly from the device (e.g. direct trigger via a light switch). The output parameter RD_DONE signalizes whether the data has been received.

| Parameter | Declaration | Data type | Description |
|---|---|---|---|
| ReadingResult. nCounter | Output | BYTE | The receipt counter is incremented by one as soon as a new read result has been received.<br><br>Value area:<br>[0x00..0xFF] |
| ReadingResult. iLength | Output | INT | Byte length of the receiving read result. |
| ReadingResult. arrResult | Output | ARRAY [1..200] of BYTE | Receiving answer of a trigger signal (can be defined via the SOPAS output format).<br><br>The maximal length of the receiving data is 200 Bytes. Chapter 4.6 describes the procedure when receiving longer data telegrams. |

*Table 3: Reading Result Parameter*

## 4.6 Receipt of read results > 200 Byte

The function block is laid out to receive read results up to a length of 200 Bytes. If longer data has to be received, the function block has to be changed at the following positions:

Change in SICK LECTOR CLV DATA UDT:
In the delivered UDT (DB72) the length of the array „ReadingResult.
arrResult“ has to be adapted in such a way, that the read result which has to be received fits into the data area of the variable.

| | | | | |
|---|---|---|---|---|
| +426.0 | ReadingResult | STRUCT | | -- READING RESULT -- |
| +0.0 | nCounter | BYTE | B#16#0 | This counter is incremented if a new reading result has arrived (OUT) |
| +2.0 | iLength | INT | 0 | Byte length of the reading result (OUT) |
| +4.0 | arrResult | ARRAY[1..200] | | Reading result data (OUT) |
| *1.0 | | CHAR | | |
| =204.0 | | END_STRUCT | | |

*Image 11: Receipt of read results > 200 Bytes (change in the UDT)*

Change in SICK LECTOR CLV TCP CP UDT:
In the static area of the variable survey, the length of the variable „arrRecord“ has to be adapted in such a way, that the read result fits into the data area of the variable. The array must not be below a length of 250 bytes, but it has to be larger or equal to the length of the ReadingResult.arrResult.

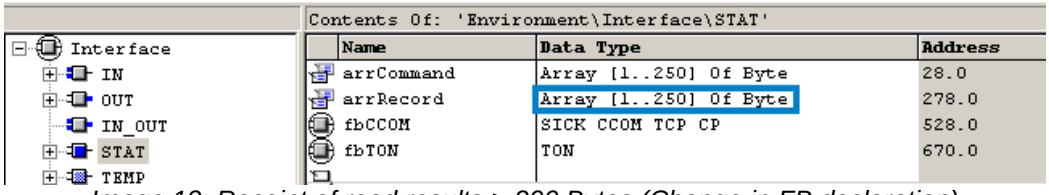



*Image 12: Receipt of read results > 200 Bytes (Change in FB declaration)*

The new defined lengths of the array have to be put into network 3 of SICK LECTOR CLV TCP CP FBs.

```
Network 3: CONFIGURATION
- Configure the length of the "Record" array
- Configure the length of the "Command" array
- Configure the length of the "Reading Result" array
- Configure [STX]/[ETX] framing flag

PLEASE NOTE:
"Record" array >= "Command" array
"Record" array >= "Reading Result" array

//-- LENGTH OF THE RECORD ARRAY
      L     250
      T     #iArrayRecLen                #iArrayRecLen

//-- LENGTH OF THE COMMAND ARRAY
      L     250
      T     #iArrayComLen                #iArrayComLen

//-- LENGTH OF THE READING RESULT ARRAY
      L     200
      T     #iArrayReadLen               #iArrayReadLen

//-- FRAMING
      CLR
      =     #bAddFraming                 #bAddFraming

//-- RESET READING RESULT FLAG
      CLR
      =     #RD_DONE                     #RD_DONE
```

*Image 13: Receipt of read results > 200 Bytes (change in the code of the function block)*

After the change the instance of the function block has to be actualized. Afterwards the changed UDT and the function block have to be transferred together with the updated instance to the PLC.

# 5 Parameter

| Parameter | Declaration | Data type | Storing area | Description |
|---|---|---|---|---|
| EN | INPUT | BOOL | E,M,D,L, Konst. | Enable entry (KOP and FUP) |
| ID | INPUT | INT | E,M,D,L, Konst. | Connection-ID to the projected TCP-connection (see NetPro connection setting Image 4). |
| LADDR | INPUT | WORD | E,M,D,L, Konst. | Component entry address of the CP-module (see NetPro connection setting Image 4). |
| CAN_ID | INPUT | INT | E,M,D,L, Konst. | CAN-ID of the sensor to be contacted.<br><br>If no CAN-network is used, the CAN-ID = 0.<br><br>The master resp. the multiplexer is always contacted with CAN-ID = 0, even if another CAN-ID is assigned. |
| TOUT | INPUT | TIME | E,M,D,L, Konst. | Time after which a timeout error is provoked. |
| START_REQ | INPUT | BOOL | E,M,D,L | Positive edge:<br>Carrying out the selected function block action. |
| TRIG_ON | INPUT | BOOL | E,M,D,L, Konst. | Function block action: Carrying out a device trigger (open trigger window) |
| TRIG_OFF | INPUT | BOOL | E,M,D,L, Konst. | Function block action: Carrying out a device trigger (close trigger window)<br><br>The from the device sent result (SOPAS output format) is stored in the variable „ReadingResult.sResult“ of the transferring data structure (DB72). |
| MATCH_ CODE | INPUT | BOOL | E,M,D,L Konst. | Creating a Matchcode condition.<br><br>This requires that the parameters described in chapter 4.5.1 are set in the structure (Matchcode). |
| COM_TEST | INPUT | BOOL | E,M,D,L, Konst. | Function block action: Carrying out a communication test.<br><br>REQ_DONE= TRUE:<br>Communication OK<br><br>REQ_DONE= FALSE:<br>Communication not OK |
| SAVE_PER MANENT | INPUT | BOOL | E,M,D,L, Konst. | Permanent saving of all device parameters in the device. |

| Parameter | Declara-tion | Data type | Storing area | Description |
|---|---|---|---|---|
| FREE_ COMMAND | INPUT | BOOL | E,M,D,L, Konst. | Function block action: Carrying out a free command.<br><br>This requires that the UDTs (DB72) in the structure (FreeCommand) as well as the parameters iCommandLength and arrCommand contain valid data (see chapter 4.5.2).<br><br>After a successful transfer (bReqDone=TRUE) the command reply is available in the data structure. |
| DATA | INPUT | BLOCK_ DB | Konst. | Transfer of the respective UDT which is necessary for the configuration of the function block and for storing the read results (DB72). |
| RD_DONE | OUTPUT | BOOL | A,M,D,L | Positive edge:<br>New read result is received. The con-tent of the read result can be configured with SOPAS-ET (see chapter 3.3). |
| REQ_DONE | OUTPUT | BOOL | A,M,D,L | Indicates if the chosen function block action can be carried out without error.<br><br>TRUE: processing terminated<br>FALSE: processing not terminated |
| REQ_BUSY | OUTPUT | BOOL | A,M,D,L | Request is in process. |
| ERROR | OUTPUT | BOOL | A,M,D,L | Error bit:<br><br>0: No error<br>1: Break-off with error |
| ERROR CODE | OUTPUT | WORD | A,M,D,L | Error status (see error codes) |
| ENO | OUTPUT | BOOL | A,M,D,L | Enable output (KOP and FUP) |

*Table 4: Function block parameter*

# 6 Error Codes

The parameter ERRORCODE contains the following error information:

| Error code | Short description | Description |
|---|---|---|
| W#16#0000 | No error | No error |
| W#16#0001 | Timeout error | Order has not been finished within the chosen timeout.<br><br>This could be because of:<br>- Device is not connected with PLC<br>- CAN-Bus participant is not available<br>- Wrong communication parameter |
| W#16#0002 | Internal function block error | Internal function block error |
| W#16#0003 | No or more than one function block action selected | Only one function block action can be carried out at the same time |
| W#16#0004 | Received read result > Reading Result Array | The received read result is longer than 200 Bytes. For the receipt of longer read results, please have a look at chapter 4.6 |
| W#16#0005 | 100 < FreeCommand. iCommandLength <=0 | Invalid length of the free command<br><br>Valid value area:<br>[1…100] |
| W#16#0006 | Answer of the free command > 100 Byte | The answer to the sent free command is longer than 100 Byte. |
| W#16#0007 | 63 < CAN_ID < 0 | Invalid CAN-ID<br><br>Valid value area:<br>[0..63] |
| W#16#0008 | Reserved | Reserved |
| W#16#0009 | Communication error | Communication to the device cannot be realized.<br><br>This could be because of:<br>- Invalid ID parameter<br>- No connection has been set up<br>- A telegram > arrRecord has been received |
| W#16#**XX**0A | Device error | A device error has come up (’sFA XX’)<br><br>**XX** = device error (see device documentation) |
| W#16#000B | Invalid Command answer | The selected action has not been carried out.<br><br>This could be because of:<br>- Wrong trigger setting in the SOPAS object<br>- Device is not in „Run-Mode“ |
| W#16#000C – W#16#000F | Reserved | Reserved |

| Error code | Short description | Description |
|---|---|---|
| W#16#0010 | 9 < Match-code.nMatchNumber <= 0 | Invalid Matchcode number.<br><br>Valid value area:<br>[1..9] |
| W#16#0011 | 75 < Matchcode.<br>iLength <= 0 | Invalid Matchcode length.<br><br>Valid value area:<br>[1..75] |
| W#16#**XX**12 | Matchcode / Save Perma-nently have not been car-ried out. | The selected action has not been carried out. The device has put into „RUN-Mode“ once again.<br><br>**XX** = Error code of the shown errors |
| W#16#0013 | Change into „RUN-Mode“ is not possible | The device cannot be put back into the „RUN-Mode“.<br><br>This could be because of:<br>- Command needs more time than the set Timeout of the routine. |

*Table 5: Error Code*

# 7 Example

Image 14 shows an example of a circuit of SICK LECTOR CLV TCP CP FBs. The TCP connection to the SICK Sensor has been projected via NetPro before and the connection parameters (ID) and (LADDR) have been transferred to the function block (see chapter 3.2). Since the sensor is not in a CAN network, a cero is put as CAN-ID.

For the trigger result (defined in SOPAS Output format) 200 Bytes are reserved in DB72. If the read result is longer, this is indicated by an error at the function block.

**Program selection:**

```
Network 2: CALL SICK LECTOR CLV FUNCTION BLOCK
Comment:

      CALL  "SICK LECTOR CLV TCP CP" , "INSTANCE_FB72"     FB72 / DB172
       ID           :=1
       LADDR        :=W#16#100
       CAN_ID       :=0
       TOUT         :=T#10S
       START_REQ    :="bRequest"                          M10.0
       TRIG_ON      :="bTriggerOn"                        M12.1
       TRIG_OFF     :="bTriggerOff"                       M12.2
       MATCH_CODE   :="bMatchCode"                        M12.3
       COM_TEST     :="bComTest"                          M12.4
       SAVE_PERM    :="bSavePermanent"                    M12.5
       FREE_COMMAND:="bFreeCommand"                       M12.6
       DATA         :="SICK LECTOR CLV6XX DATA"           DB72
       RD_DONE      :="bRdDone"                           M10.1
       REQ_DONE     :="bReqDone"                          M10.2
       REQ_BUSY     :="bReqBusy"                          M10.3
       ERROR        :="bError"                            M10.4
       ERRORCODE    :="nErrorcode"                        MW14
```

*Image 14: Example of a circuit of SICK LECTOR CLV TCP CP FBs*

## 7.1 Create / change matchcode

In order to create a new matchcode evaluation condition or to change an existing one, the necessary parameter values have to be put first.

nMatchNumber: Matchcode Number (here: Match5)
nCodeType: Code Type (“*“= All Code Types)
nLength: Length of the code
iContentLength: Code content

```
// ========== Match Code ==========
DB72.DBB   0   "SICK LECTOR CLV6XX DATA".Matchcode.nMatchNumber      DEC         5
DB72.DBB   1   "SICK LECTOR CLV6XX DATA".Matchcode.nCodeType        CHARACTER   '*'
DB72.DBB   2   "SICK LECTOR CLV6XX DATA".Matchcode.nLength          DEC         5
DB72.DBW   4   "SICK LECTOR CLV6XX DATA".Matchcode.iContentLength   DEC         4
DB72.DBB   6   "SICK LECTOR CLV6XX DATA".Matchcode.arrContent[1]    CHARACTER   'A'
DB72.DBB   7   "SICK LECTOR CLV6XX DATA".Matchcode.arrContent[2]    CHARACTER   'B'
DB72.DBB   8   "SICK LECTOR CLV6XX DATA".Matchcode.arrContent[3]    CHARACTER   'C'
DB72.DBB   9   "SICK LECTOR CLV6XX DATA".Matchcode.arrContent[4]    CHARACTER   '*'
DB72.DBB   10  "SICK LECTOR CLV6XX DATA".Matchcode.arrContent[5]    CHARACTER   ' '
```

*Image 15: Matchcode Parameter*

The matchcode action *bMatchcode* is carried out as soon as the bit *bRequest* is triggered with a positive edge.

| // SICK Lector / CLV Function Block Example | | | |
|---|---|---|---|
| MW 16 | "iCanID" | DEC | 0 |
| M 10.0 | "bRequest" | BOOL | true |
| | | | |
| M 10.2 | "bReqDone" | BOOL | true |
| M 10.3 | "bReqBusy" | BOOL | false |
| M 10.4 | "bError" | BOOL | false |
| MW 14 | "nErrorcode" | HEX | W#16#0000 |
| | | | |
| // Selection of the FB action to be execute | | | |
| M 12.1 | "bTriggerOn" | BOOL | false |
| M 12.2 | "bTriggerOff" | BOOL | false |
| M 12.3 | "bMatchCode" | BOOL | true |
| M 12.4 | "bComTest" | BOOL | false |
| M 12.5 | "bSavePermanent" | BOOL | false |
| M 12.6 | "bFreeCommand" | BOOL | false |

*Image 16: Start matchcode action*

The matchcode action is finished das soon as „bReqDone = TRUE“. This example puts the following evaluation condition on the device.

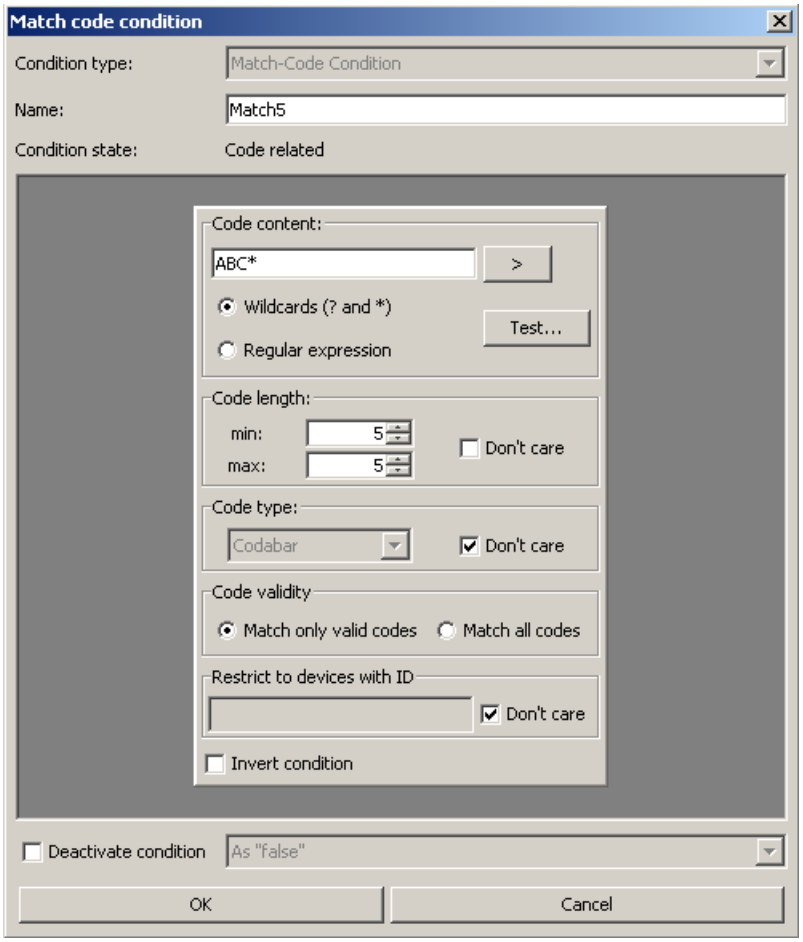



*Image 17: Put evaluation condition*

## 7.2 Send trigger signal

That a trigger via the PLC can take place, the trigger source has to be set before to „command“ via SOPAS-ET.

In this example the reading gate can be opened and closed via the function block. Optionally, the reading gate will be closed automatically in the case of a „Good Read“.

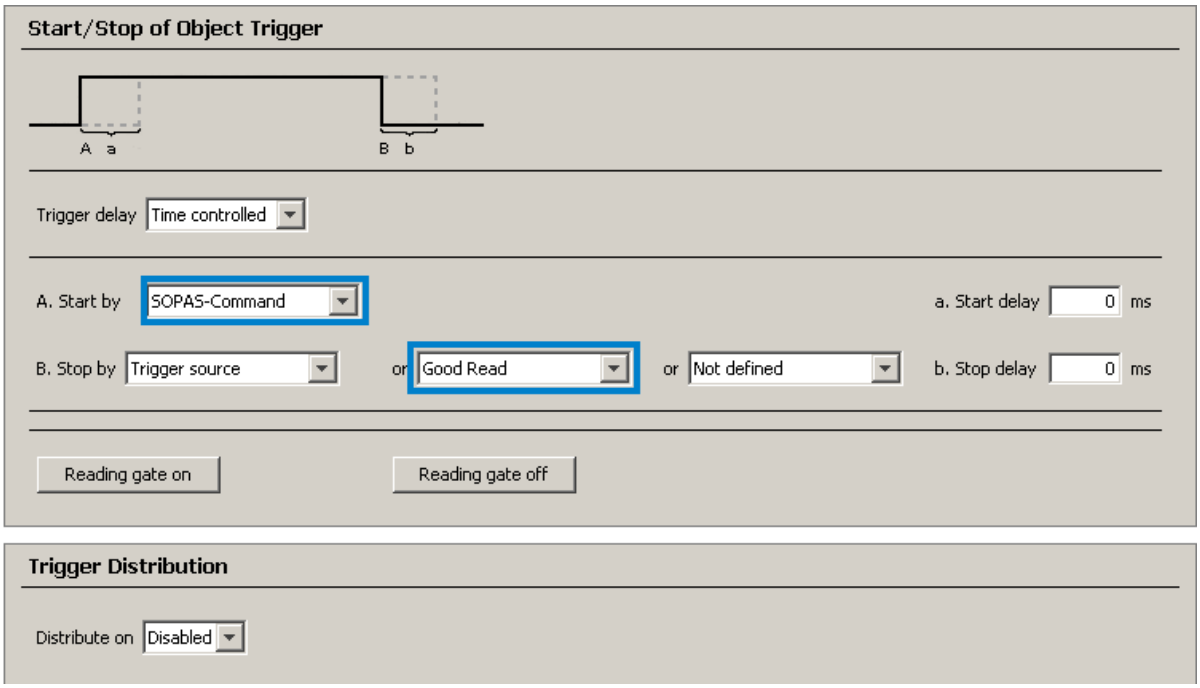



*Image 18: SOPAS object trigger settings*

This is done, as soon as the bit „bRequest“ is triggered with a positive edge. The reading gate is open as soon as „bReqDone = TRUE.

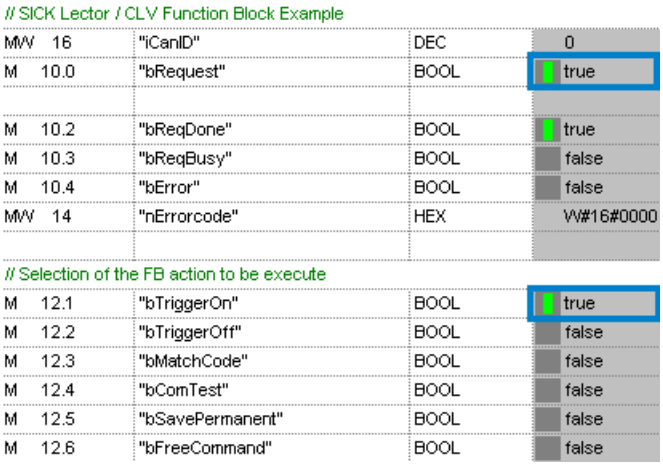



*Image 19: Open reading gate*

If the code has been read successfully („Good Read“), the device closes the reading gate automatically and sends the read code to the PLC. The function block saves the read code in the array „ReadingResult.arrResult“ of the UDT (DB72). The parameter RD_DONE indicates

for one PLC cycle, that new data has been received. The parameter ReadingResult.iLength indicates, how many bytes have been received resp. how many are valid.

// ========== Reading Result ==========

| | | | |
|---|---|---|---|
| DB72.DBB 286 | "SICK LECTOR CLV6XX DATA".ReadingResult.iCounter | HEX | B#16#18 |
| DB72.DBW 288 | "SICK LECTOR CLV6XX DATA".ReadingResult.iLength | DEC | 16 |
| DB72.DBB 290 | "SICK LECTOR CLV6XX DATA".ReadingResult.arrResult[1] | CHARACTER | 'ך' |
| DB72.DBB 291 | "SICK LECTOR CLV6XX DATA".ReadingResult.arrResult[2] | CHARACTER | 'S' |
| DB72.DBB 292 | "SICK LECTOR CLV6XX DATA".ReadingResult.arrResult[3] | CHARACTER | 'I' |
| DB72.DBB 293 | "SICK LECTOR CLV6XX DATA".ReadingResult.arrResult[4] | CHARACTER | 'C' |
| DB72.DBB 294 | "SICK LECTOR CLV6XX DATA".ReadingResult.arrResult[5] | CHARACTER | 'K' |
| DB72.DBB 295 | "SICK LECTOR CLV6XX DATA".ReadingResult.arrResult[6] | CHARACTER | ' ' |
| DB72.DBB 296 | "SICK LECTOR CLV6XX DATA".ReadingResult.arrResult[7] | CHARACTER | 'T' |
| DB72.DBB 297 | "SICK LECTOR CLV6XX DATA".ReadingResult.arrResult[8] | CHARACTER | 'e' |
| DB72.DBB 298 | "SICK LECTOR CLV6XX DATA".ReadingResult.arrResult[9] | CHARACTER | 's' |
| DB72.DBB 299 | "SICK LECTOR CLV6XX DATA".ReadingResult.arrResult[10] | CHARACTER | 't' |
| DB72.DBB 300 | "SICK LECTOR CLV6XX DATA".ReadingResult.arrResult[11] | CHARACTER | ' ' |
| DB72.DBB 301 | "SICK LECTOR CLV6XX DATA".ReadingResult.arrResult[12] | CHARACTER | 'C' |
| DB72.DBB 302 | "SICK LECTOR CLV6XX DATA".ReadingResult.arrResult[13] | CHARACTER | 'o' |
| DB72.DBB 303 | "SICK LECTOR CLV6XX DATA".ReadingResult.arrResult[14] | CHARACTER | 'd' |
| DB72.DBB 304 | "SICK LECTOR CLV6XX DATA".ReadingResult.arrResult[15] | CHARACTER | 'e' |
| DB72.DBB 305 | "SICK LECTOR CLV6XX DATA".ReadingResult.arrResult[16] | CHARACTER | 'ן' |
| DB72.DBB 306 | "SICK LECTOR CLV6XX DATA".ReadingResult.arrResult[17] | CHARACTER | ' ' |

*Image 20: Read result*