

SICK LECTOR / CLV6xx Funktionsbaustein

Bausteinversion V2.X

SICK LECTOR CLV TCP CP Funktionsbaustein für
Siemens Step7 Steuerungen



Inhaltsverzeichnis

1 Zu diesem Dokument	3
1.1 Funktion dieses Dokuments	3
1.2 Zielgruppe.....	3
2 Allgemeines	4
3 Hardwarekonfiguration	5
3.1 Unterstützte SPS-Steuerungen	5
3.2 Konfiguration in Step7	5
3.3 SOPAS Gerätekonfiguration	9
4 Bausteinbeschreibung	11
4.1 Bausteinspezifikationen	11
4.2 Arbeitsweise	12
4.3 Verhalten im Fehlerfall	13
4.4 Timing.....	13
4.5 Werteübergabe	14
4.5.1 Matchcode.....	15
4.5.2 Free Command	16
4.5.3 Reading Result.....	16
4.6 Empfangen von Leseergebnissen > 200 Byte	17
5 Parameter	19
6 Fehlercodes	21
7 Beispiel	23
7.1 Matchcode ändern/anlegen.....	23
7.2 Triggersignal senden	25

1 Zu diesem Dokument

Bitte lesen Sie dieses Kapitel sorgfältig, bevor Sie mit dieser Betriebsanleitung und den SICK Lector/CLV6xx Funktionsbaustein arbeiten.

1.1 Funktion dieses Dokuments

Diese Betriebsanleitung beschreibt den Umgang mit dem SICK LECTOR CLV6XX TCP CP Funktionsbaustein. Sie leitet das technische Personal des Maschinenherstellers bzw. Maschinenbetreibers zur Projektierung und Inbetriebnahme des Funktionsbausteins an.

1.2 Zielgruppe

Diese Betriebsanleitung richtet sich an fachkundiges Personal wie z.B. Techniker oder Ingenieure.

2 Allgemeines

Der Funktionsbaustein „SICK LECTOR CLV TCP CP“ wird zur Kommunikation zwischen einer SIMATIC Steuerung und einem SICK Lector 2D-Codeleser bzw. einem CLV6xx Barcodeleser verwendet. Der Code Reader kommuniziert über eine TCP-Verbindung mit der Steuerung.

Die folgende Abbildung zeigt die Darstellung des Funktionsbausteins in der Funktionsplan-Ansicht (FUP).

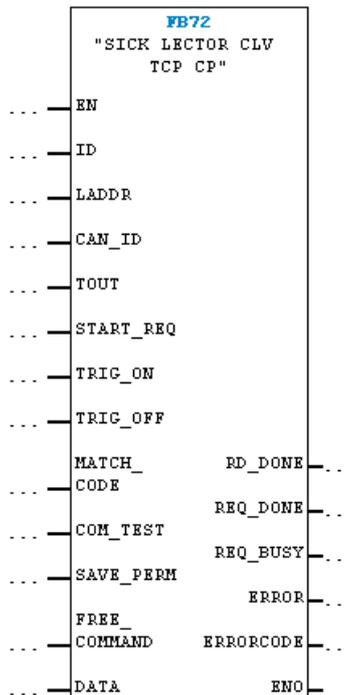


Abbildung 1: SICK LECTOR CLV TCP CP Funktionsbaustein

Bausteinfunctionalitäten:

- Senden eines Triggerbefehls (CoLaⁱ Kommando) über die SPS
- Empfang von Leseergebnissen (im SOPAS-ETⁱⁱ Ausgabeformat definiert)
- Anlegen / Ändern einer Evaluationsbedingung für einen Matchcode
- Ausführen eines Kommunikationstests
- Permanentes Speichern aller Geräteparameter im Gerät
- Kommunikation über frei wählbare CoLa Kommandos (CoLa-A Protokoll)
- Ansprechen von Geräten, die untereinander via CAN-Bus kommunizieren

ⁱ Die Command Language (CoLa) ist ein internes SICK Protokoll zur Kommunikation mit SOPAS-Geräten

ⁱⁱ SOPAS-ET ist ein Engineering Tool zum parametrieren von SICK Sensoren

3 Hardwarekonfiguration

3.1 Unterstützte SPS-Steuerungen

Der Funktionsbaustein darf nur mit Simatic S7-Steuerungen der 300er Familie betrieben werden. Es werden nur Steuerungen unterstützt, die für die TCP Kommunikation ein CP-Modul verwenden. Steuerungen mit integrierter Ethernet-Schnittstelle werden nicht unterstützt.

3.2 Konfiguration in Step7

Bevor der Funktionsbaustein verwendet werden kann, muss in eine TCP-Verbindung zum Sensor hergestellt werden. Öffnen Sie hierfür das NetPro Verbindungstool im Simatic Manager (Simatic Manager → Extras → Netz konfigurieren).

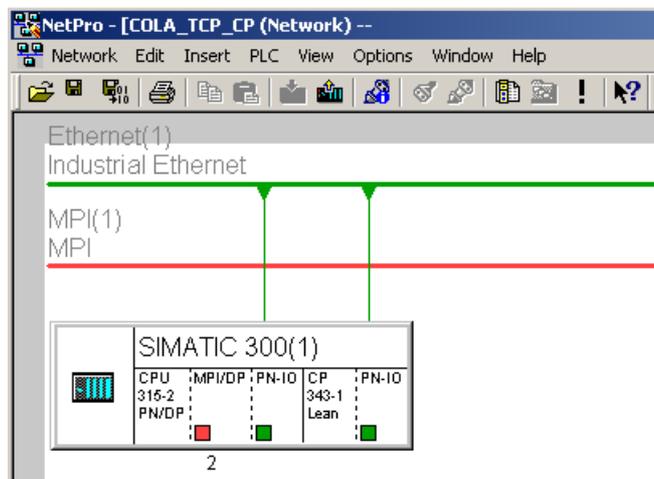


Abbildung 2: NetPro (Step7)

Markieren Sie die CPU in ihrer S7-Station legen sie unter „Einfügen → Neue Verbindung“ eine unspezifizierte TCP-Verbindung an.

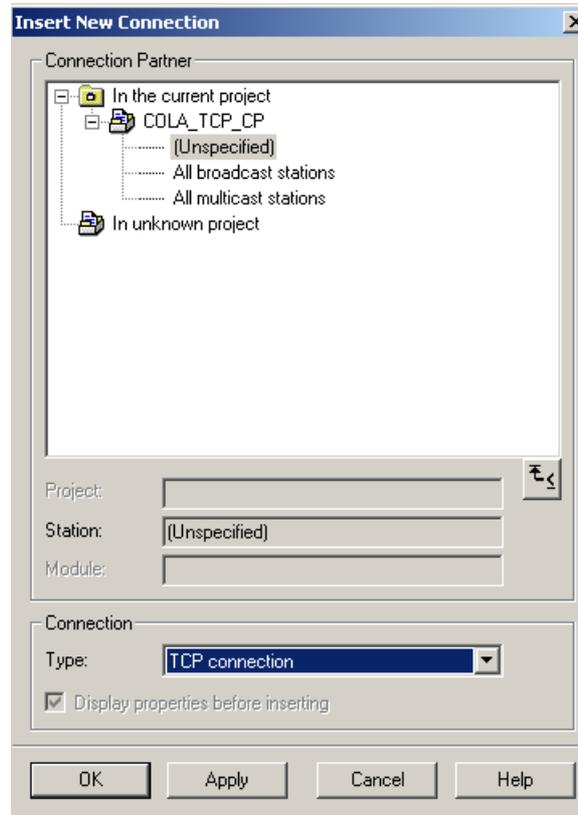


Abbildung 3: Anlegen einer TCP-Verbindung in NetPro (Step7)

Im Eigenschaftsdialog der TCP-Verbindung muss im Register „Allgemein“ die Checkbox „Aktiver Verbindungsaufbau“ aktiviert werden. Auf der Rechten Seite im Dialog werden die Verbindungsparameter für die TCP-Verbindung angezeigt. Diese Parameterwerte müssen dem SICK LECTOR CLV TCP CP Funktionsbaustein beim Bausteinaufruf übergeben werden (Parameter: ID / LADDR).

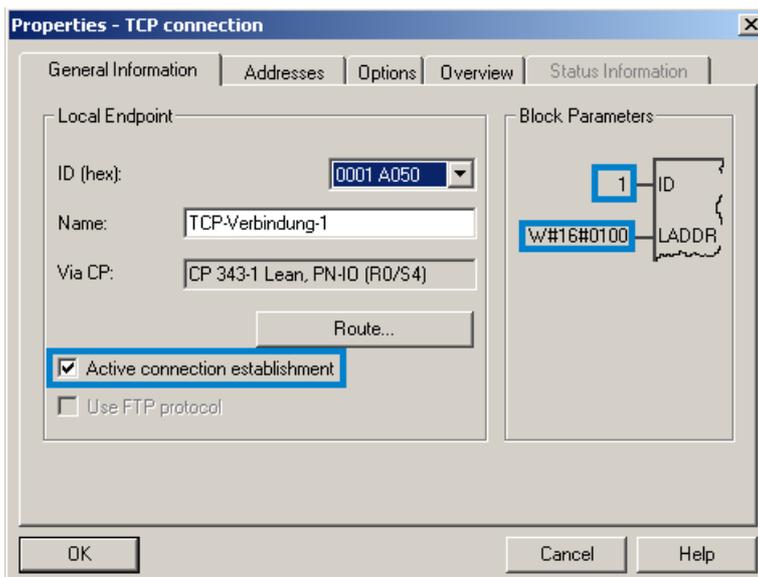


Abbildung 4: NetPro TCP Verbindungseinstellungen (Step7)

Im Register „Adressen“ muss die IP-Adresse und der verwendete Port des SICK Gerätes angegeben werden. Standardmäßig verwenden SICK-Sensoren zum kommunizieren den Port 2112.

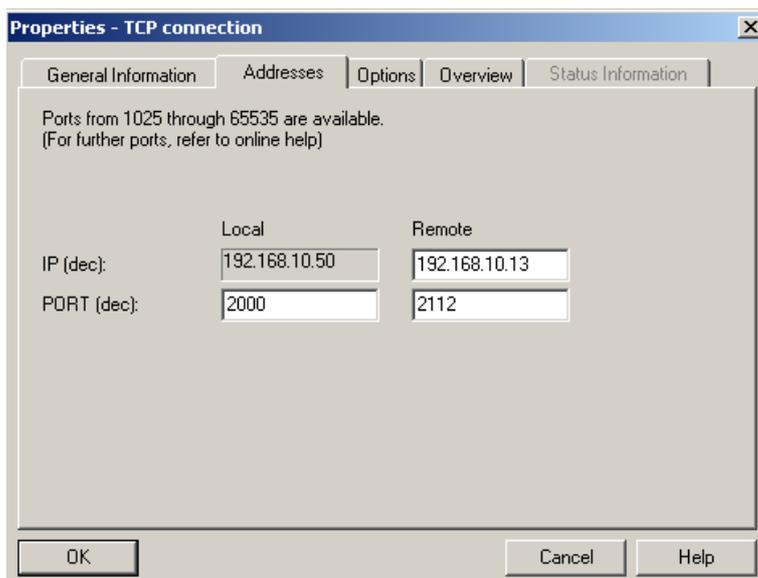


Abbildung 5: Adressen der TCP-Verbindung (Step7)

Im Register „Optionen“ ist die Betriebsart „Send/Recv“ einzustellen.

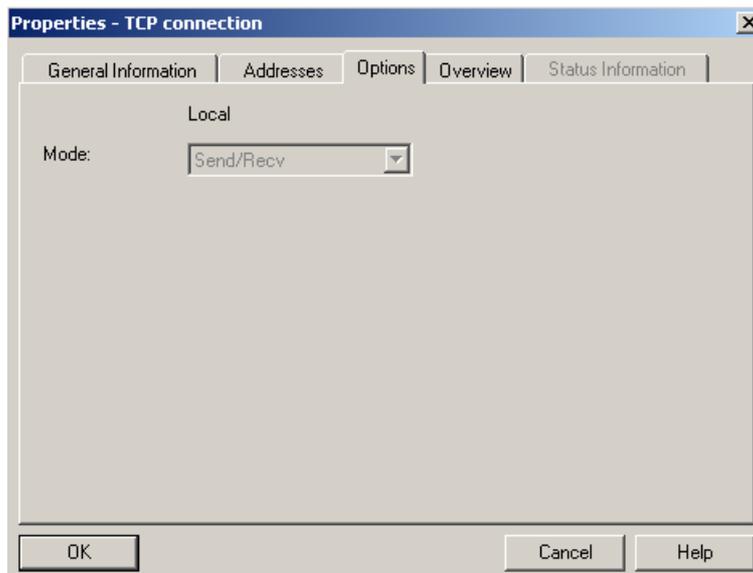


Abbildung 6: Betriebsart der TCP-Verbindung (Step7)

Nachdem der Eigenschaftsdialog mit „OK“ geschlossen wird, wird automatisch eine TCP-Verbindung in der Verbindungstabelle angezeigt. Speichern und übersetzen Sie nun die Station und laden Sie anschließend die Verbindung in Ihre S7-Steuerung. Ggf. ist ein Neustart des Sensors nötig, um die TCP-Verbindung herzustellen.

Um die projektierte TCP-Verbindung zu diagnostizieren kann unter „Zielsystem → Verbindungsstatus aktivieren“ der Verbindungsstatus abgefragt werden.

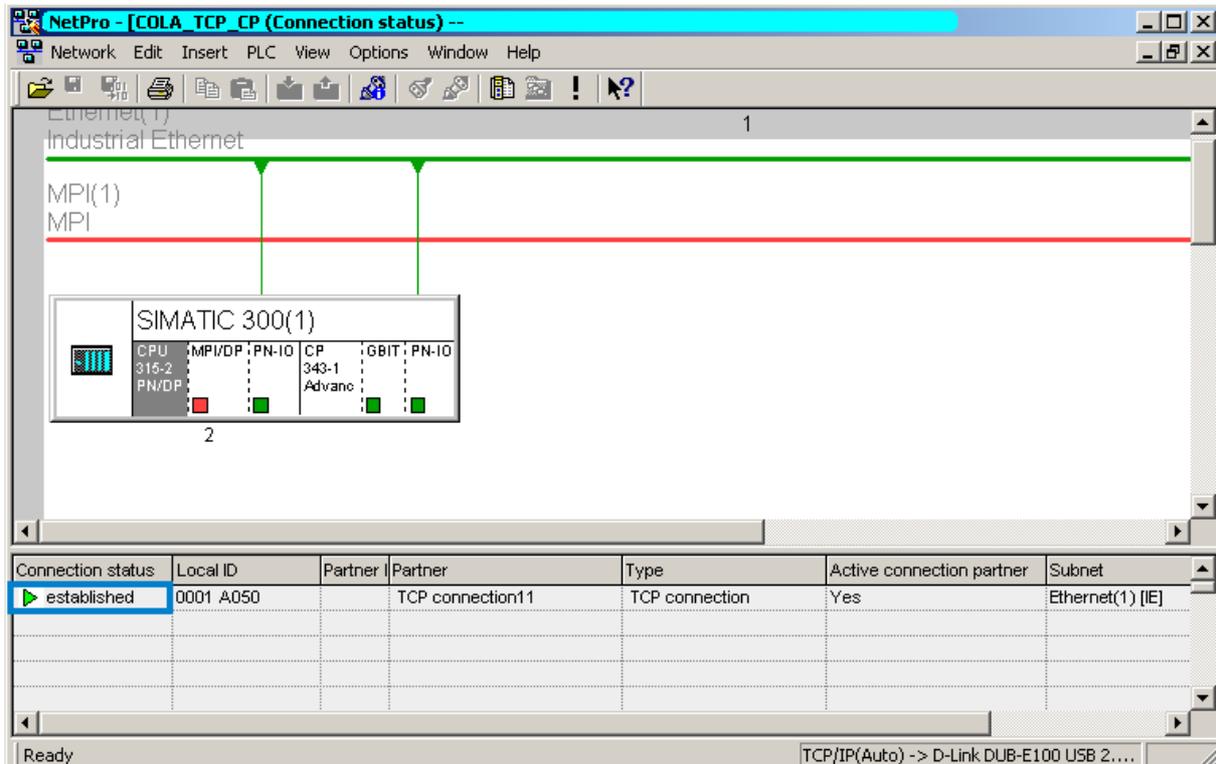


Abbildung 7: Verbindungsstatus der TCP-Verbindung (Step7)

3.3 SOPAS Gerätekonfiguration

Damit eine Triggerung über die SPS erfolgen kann, muss die Triggerquelle zuvor mit SOPAS-ET auf „Kommando“ eingestellt werden. Abbildung 8 zeigt, wie unter dem Menüpunkt „Objekt Trigger“ der CLV6xx bzw. der Lector parametrierung wird.

- Start mit "SOPAS-Kommando" (TRIG_ON Kommando muss verwendet werden)
 - Stop mit "SOPAS-Kommando" (TRIG_OFF Kommando muss verwendet werden)
- Optional kann das Triggerfenster auch automatisch geschlossen werden, wenn der Sensor einen Code gelesen hat „Good Read“ oder im Falle eines „No Reads“ nach einem definierten Timeout (hier 500ms).

Abbildung 8: SOPAS Trigger Einstellungen

Sollte das Gerät direkt getriggert werden z.B. über eine Lichtschranke oder über ein Hardware-signal am Sensor1-Eingang des LECTOR/CLV können die Baustein-funktionen TRIG_ON / TRIG_OFF nicht mehr verwendet werden. Wenn ein Triggerergebnis vom Bau-stein empfangen wird, wird dies immer über Ausgangsparameter „RD_DONE“ signalisiert.

Die Leseergebnisdaten müssen immer von SOPAS Kommandos unterscheidbar sein. Wenn das Lesergebnis ggf. auch mit „s“ (Kleinbuchstabe „s“ so wie ein SOPAS Kommando) beginnen kann, so ist das Ausgabeformat z.B. durch voranstellen eines „D“ zu verändern.

Abbildung 9: SOPAS Ausgabeformat Einstellungen

4 Bausteinbeschreibung

Der Funktionsbaustein ist ein asynchron arbeitender FB, d. h. die Bearbeitung erstreckt sich über mehrere FB-Aufrufe. Dies setzt voraus, dass der Baustein zyklisch im Anwenderprogramm aufgerufen wird.

Der Baustein kapselt den Funktionsbaustein „SICK CCOM TCP CP“ (FB11), der die Kommunikation zwischen SPS und Sensor ermöglicht.

4.1 Bausteinspezifikationen

Bausteinnummer:	FB72
Bausteinname:	SICK LECTOR CLV TCP CP
Version:	2.0
Aufgerufene Bausteine:	FC5 (AG_SEND) FC6 (AG_RECV) SFC20 (BLKMOV) SFB4 (TON) FB11 (SICK CCOM TCP CP) FB103 (AG_RECV_TCP_xVAR)
Verwendete Datenbausteine:	DB72 (SICK LECTOR CLV DATA)
Bausteinanruf:	Zyklisch
Verwendete Merker:	keine
Verwendete Zähler:	keine
Verwendetes Register:	AR1, AR2 (für Multiinstanzaufruf)
Multiinstanzfähig:	ja
Erstelsprache:	Step7-AWL
Step7 Version:	Simatic Step7 V5.5

Die im Funktionsbaustein verwendeten Systemfunktionen (SFCs) müssen auf der jeweils verwendeten Steuerung vorhanden sein.

Beim ändern von Bausteinnummern müssen die entsprechenden Aufrufe im SICK LECTOR CLV TCP CP Baustein aktualisiert werden.

4.2 Arbeitsweise

Um den Funktionsbaustein einsetzen zu können, müssen zunächst die folgenden Kommunikationsparameter angegeben werden:

ID: Verbindungs-ID der TCP-Verbindung. Der Parameter wird in den NetPro Verbindungseigenschaften angezeigt (siehe Abbildung 4).

LADDR: Anfangsadresse der CP-Baugruppe. Der Parameter wird in den NetPro Verbindungseigenschaften angezeigt (siehe Abbildung 4).

DATA: Der zum Funktionsbaustein gehörende Datenbaustein (DB72) beinhaltet Ein- und Ausgabeparameter der unterstützten Bausteinaktionen. Der Datenbaustein muss dem Eingangsparameter „DATA“ des Funktionsbausteins übergeben werden.

Ausführbare Bausteinaktionen:

- Trigger on → Öffnet das Lesetor des Gerätes per CoLa Kommando
- Trigger off → Schließt das Lesetor des Gerätes per CoLa Kommando
- Match Code → Erstellt / Ändert eine neue Evaluationsbedingung für einen Matchcode
- Kommunikationstest → Prüft, ob das Gerät per „sRIO“ Kommando erreichbar ist
- Save Permanent → Speichert alle Geräteparameter dauerhaft im Gerät ab.
- Free Command → Ausführen eines frei wählbaren CoLa Kommandos

Um eine Bausteinaktion (TRIG_ON, TRIG_OFF, etc.) auszuführen, muss zunächst die gewünschte Aktion ausgewählt werden. Es kann immer nur eine Aktion gleichzeitig ausgeführt werden. Um die Aktion auszuführen, muss der Parameter START_REQ mit einer positiven Flanke (Signalwechsel von logisch null auf eins) angetriggert werden. Solange noch keine gültige Geräteantwort empfangen wurde, wird dies über den Parameter REQ_BUSY signalisiert.

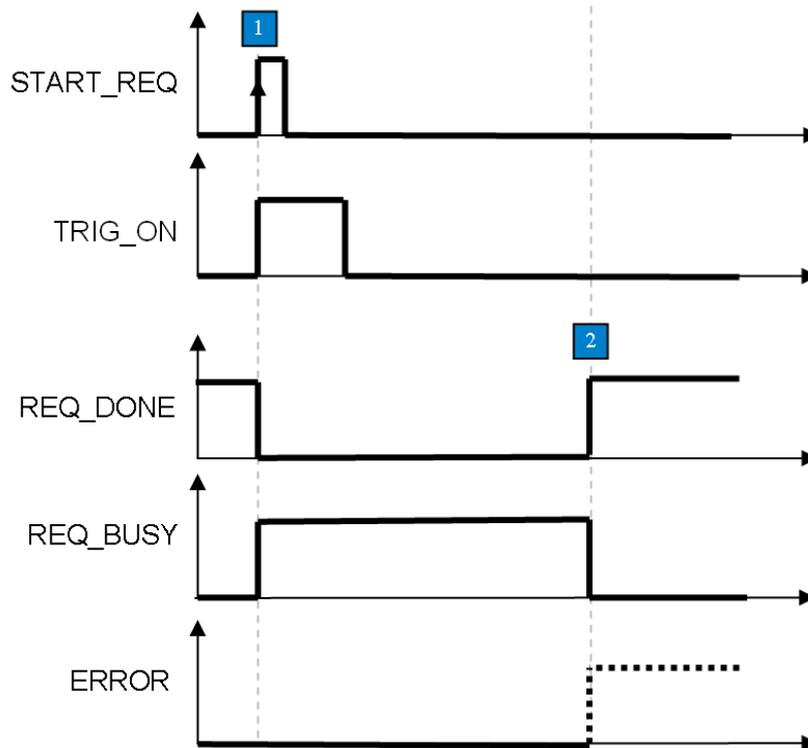
Wenn der Baustein am Ausgangsparameter REQ_DONE = TRUE signalisiert, wurde die Aktion erfolgreich durchgeführt. Wurden bei dieser Aktion (z.B. FREE_COMMAND) Daten vom Gerät angefordert, werden diese in den jeweiligen Datenbereich des zugehörigen Nutzdatenbausteins (DATA) kopiert.

Daten die per Triggerbefehl (TRIG_ON, TRIG_OFF) oder direkt vom Gerät gesendet werden (z.B. direkter Trigger über eine Lichtschranke), werden in den Datenbaustein (ReadingResult.arrResult) abgelegt. Der Ausgangsparameter RD_DONE zeigt für einen SPS Zyklus an, dass neue Daten empfangen wurden. Die vom Gerät gesendeten Daten können im SOPAS Ausgabeformat geändert, bzw. angepasst werden (siehe Kapitel 3.3).

4.3 Verhalten im Fehlerfall

Bei einem fehlerhaften Eingabewert oder einer fehlerhaften Eingangsbeschaltung des FBs, wird ein Errorbit (ERROR) gesetzt und ein Fehlercode (ERRORCODE) ausgegeben. In diesem Fall wird keine weitere Bearbeitung durchgeführt. Die Diagnoseparameter (ERROR, ERRORCODE) des FBs behalten solange ihren Wert, bis ein neuer Auftrag gestartet wird.

4.4 Timing



1: Anforderung durch Pos Flanke an START_REQ

Die gewünschte Aktion (hier TRIG_ON) muss vorher/zeitgleich ausgewählt werden. Es darf nur eine Aktion zeitgleich ausgewählt werden, sonst wird mit „ERROR“ abgebrochen.

2: Wenn alle Kommandos gesendet sind und alle Antworten empfangen wurden, wird die Aktion mit „REQ_Done“ beendet. Wenn die Aktion fehlerhaft verläuft, wird mit „ERROR“ beendet. Bei Abbruch mit „ERROR“ enthält „ERRORCODE“ den aufgetretenen Fehler.

4.5 Werteübergabe

Der mitgelieferte Datenbaustein „SICK LECTOR CLV DATA“ (DB72) beinhaltet Ein- und Ausgabeparameter aller unterstützten Bausteinaktionen. Der Datenbaustein kann je nach Anwenderprogramm umbenannt werden. Die Datenstruktur ist fest vordefiniert und darf, bis auf den letzten Eintrag (ReadingResult.arrResult), nicht geändert werden (siehe Kapitel 4.6: Empfangen von Leseergebnissen > 200 Byte).

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	Matchcode	STRUCT		-- MATCH CODE --
+0.0	nMatchNumber	BYTE	E#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	E#16#0	Sets min and max length. E#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	E#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		

Abbildung 10: Struktur des SICK LECTOR CLV DATA Nutzdaten DBs

4.5.1 Matchcode

Mit Hilfe der Matchcode Aktion hat man die Möglichkeit eine Evaluationsbedingung neu zu erstellen, oder eine bereits existierende abzuändern. Bevor die Match_Code Aktion ausgeführt wird, müssen in der Struktur Matchcode die folgenden Parameter angegeben werden.

Parameter	Deklaration	Datentyp	Beschreibung
Matchcode. iMatchNumber	Input	BYTE	Mit der Matchcode Nummer wird der Name der Evaluationsbedingung festgelegt (z.B. 1=Match1, 2=Match2). Gültiger Wertbereich: [1..9]
Matchcode. CodeType	Input	CHAR	Gewünschter Code Typ, auf der sich die Evaluationsbedingung beziehen soll (z.B. 'w'= Datamatrix, 's'= QR-Code, '*' = Beliebiger Code Typ). Eine Auswahl aller Codetypen siehe Gerätedokumentation.
Matchcode. nLength	Output	BYTE	Minimale und maximale Codelänge 0 = Beliebige Codelänge
Matchcode. iContentLength	Output	INT	Bytelänge des angegebenen Matchcodes
Matchcode. arrContent	Output	ARRAY [1..75] OF CHAR	Matchcodeinhalt

Tabelle 1: Matchcode Parameter

4.5.2 Free Command

Mit Hilfe des freien Kommandos hat man die Möglichkeit über ein gültiges CoLa Kommando mit dem Gerät zu kommunizieren. Hierfür ist es erforderlich, das Kommando in dem Parameter „arrCommand“ der Struktur „FreeCommand“ zu hinterlegen. Die Zeichenlänge des zu übertragenden Kommandos wird in den Parameter „iCommandLength“ geschrieben. Die Kommandos können der Gerätebeschreibung oder SOPAS-ET entnommen werden.

Parameter	Deklaration	Datentyp	Beschreibung
FreeCommand. iCommandLength	Input	INT	Zeichenlänge des zu übertragenden CoLa Kommandos. Gültiger Wertebereich: [1..100]
FreeCommand. arrCommand	Input	ARRAY [1..100] OF CHAR	Frei wählbares CoLa Kommando (Kommandos siehe Gerätdokumentation).
FreeCommand. iResultLength	Output	INT	Bytelänge des empfangenden CoLa Telegramms.
FreeCommand. arrResult	Output	ARRAY [1..100] OF CHAR	Empfangende Antwort des gesendeten CoLa Telegramms.

Tabelle 2: Free Command Parameter

4.5.3 Reading Result

In dem Array „ReadingResult.arrResult“ werden Daten abgelegt, die per Triggerbefehl (TRIG_ON, TRIG_OFF) oder direkt vom Gerät gesendet werden (z.B. direkter Trigger über eine Lichtschranke). Der Ausgangsparameter RD_DONE signalisiert, ob Daten empfangen wurden.

Parameter	Deklaration	Datentyp	Beschreibung
ReadingResult. nCounter	Output	BYTE	Der Empfangszähler wird um eins inkrementiert, sobald ein neues Leseergebnis empfangen wurde. Wertebereich: [0x00..0xFF]
ReadingResult. iLength	Output	INT	Bytelänge des empfangenden Leseergebnisses.
ReadingResult. arrResult	Output	ARRAY [1..200] of BYTE	Empfangende Antwort auf ein Triggersignal (Über das SOPAS Ausgabeformat definierbar). Die maximale Länge der empfangenden Daten beträgt 200 Bytes. Kapitel 4.6 beschreibt das Vorgehen beim Empfang von längeren Datentelegrammen.

Tabelle 3: Reading Result Parameter

4.6 Empfangen von Leseergebnissen > 200 Byte

Der Funktionsbaustein ist darauf ausgelegt, Leseergebnisse bis zu einer Länge von 200 Bytes zu empfangen. Sollen längere Daten empfangen werden, muss der Funktionsbaustein an den folgenden Stellen abgeändert werden:

Änderung im SICK LECTOR CLV DATA Datenbaustein:

Im mitgelieferten Nutzdatenbaustein (DB72) muss die Länge des Array „ReadingResult.arrResult“ so angepasst werden, dass das zu empfangende Leseergebnis in den Datenbereich der Variablen passt.

+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		

Abbildung 11: Empfangen von Leseergebnissen > 200 Bytes (Änderung im Datenbaustein)

Änderung im SICK LECTOR CLV TCP CP Funktionsbaustein:

Im statischen Bereich der Variablenübersicht muss die Länge der Variable „arrRecord“ so angepasst werden, dass das Leseergebnis in den Datenbereich der Variablen passt. Das Array darf eine Länge von 250 Bytes nicht unterschreiten, muss aber größer/gleich der ReadingResult.arrResult Länge sein.

Contents Of: 'Environment\Interface\STAT'		
Name	Data Type	Address
arrCommand	Array [1..250] Of Byte	28.0
arrRecord	Array [1..250] Of Byte	278.0
fbCCOM	SICK CCOM TCP CP	528.0
fbTON	TON	670.0

Abbildung 12: Empfangen von Leseergebnissen > 200 Bytes (Änderung im FB Deklaration)

Im Netzwerk 3 des SICK LECTOR CLV TCP CP FBs müssen die neu definierten Arraylängen eingetragen werden.

```
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
```

Abbildung 13: Empfangen von Leseergebnissen > 200 Bytes (Änderung im Bausteincode)

Nach der Änderung muss die Instanz des Funktionsbausteins aktualisiert werden. Anschließend muss der geänderte Nutzdatenbaustein sowie der Funktionsbaustein zusammen mit der aktualisierten Instanz neu auf die SPS übertragen werden.

5 Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN	INPUT	BOOL	E,M,D,L, Konst.	Enable Eingang (KOP und FUP)
ID	INPUT	INT	E,M,D,L, Konst.	Verbindungs-ID zur projektierten TCP-Verbindung (siehe NetPro Verbindungseinstellungen Abbildung 4).
LADDR	INPUT	WORD	E,M,D,L, Konst.	Baugruppen-Anfangsadresse des CP-Moduls (siehe NetPro Verbindungseinstellungen Abbildung 4).
CAN_ID	INPUT	INT	E,M,D,L, Konst.	CAN-ID des anzusprechenden Sensors. Wenn kein CAN-Netzwerk verwendet wird, ist die CAN-ID = 0. Der Master bzw. der Multiplexer wird immer mit der CAN-ID = 0 angesprochen, auch wenn dieser eine andere CAN-ID zugewiesen ist.
TOUT	INPUT	TIME	E,M,D,L, Konst.	Zeit, nachdem ein Timeout-Fehler ausgelöst wird.
START_REQ	INPUT	BOOL	E,M,D,L	Positive Flanke: Ausführen der gewählten Bausteinaktion.
TRIG_ON	INPUT	BOOL	E,M,D,L, Konst.	Bausteinaktion: Ausführen eines Geräte Triggers (Triggerfenster öffnen)
TRIG_OFF	INPUT	BOOL	E,M,D,L, Konst.	Bausteinaktion: Ausführen eines Geräte Triggers (Triggerfenster schließen) Das vom Gerät gesendet Ergebnis (SOPAS Ausgabeformat) wird in der Variablen „ReadingResult.arrResult“ des Nutzdaten DBs (DB72) abgelegt.
MATCH_CODE	INPUT	BOOL	E,M,D,L Konst.	Erstellen einer Matchcode Bedingung. Die Aktion setzt voraus, dass in der Struktur (Matchcode) die im Kapitel 4.5.1 beschriebenen Parameter angegeben werden.
COM_TEST	INPUT	BOOL	E,M,D,L, Konst.	Bausteinaktion: Ausführen eines Kommunikationstests. REQ_DONE= TRUE: Kommunikation OK REQ_DONE= FALSE: Kommunikation nicht OK

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
SAVE_PERMANENT	INPUT	BOOL	E,M,D,L, Konst.	Permanentes Speichern aller Geräteparameter im Gerät.
FREE_COMMAND	INPUT	BOOL	E,M,D,L, Konst.	Bausteinaktion: Ausführen eines freien Kommandos. Die Aktion setzt voraus, dass im Nutzdatenbaustein (DB72) in der Struktur (FreeCommand) die Parameter iCommandLength und arrCommand mit gültigen Daten belegt sind (siehe Kapitel 4.5.2). Die Kommandoantwort steht nach einer erfolgreichen Übertragung (REQ_DONE=TRUE) im RESULT Bereich des Datenbausteins zur Verfügung.
DATA	INPUT	BLOCK_DB	Konst.	Übergabe des zugehörigen Nutzdatenbausteins, der für die Parametrierung der Bausteinfunktionen sowie für das Ablegen des Leseergebnisses benötigt wird (DB72).
RD_DONE	OUTPUT	BOOL	A,M,D,L	Positive Flanke: Neues Leseergebnis empfangen. Der Inhalt des Leseergebnisses kann mit SOPAS-ET konfiguriert werden (siehe Kapitel 3.3).
REQ_DONE	OUTPUT	BOOL	A,M,D,L	Zeigt an, ob die gewählte Bausteinaktion fehlerfrei durchgeführt wurde. TRUE: Bearbeitung abgeschlossen FALSE: Bearbeitung nicht abgeschlossen
REQ_BUSY	OUTPUT	BOOL	A,M,D,L	Auftrag ist in Bearbeitung.
ERROR	OUTPUT	BOOL	A,M,D,L	Fehler Bit: 0: Kein Fehler 1: Abbruch mit Fehler
ERROR_CODE	OUTPUT	WORD	A,M,D,L	Fehlerstatus (siehe Fehlercodes)
ENO	OUTPUT	BOOL	A,M,D,L	Enable Ausgang (KOP und FUP)

Tabelle 4: Bausteinparameter

6 Fehlercodes

Der Parameter ERRORCODE enthält die folgenden Fehlerinformationen:

Fehlercode	Kurzbeschreibung	Beschreibung
W#16#0000	Kein Fehler	Kein Fehler
W#16#0001	Timeout Fehler	Auftrag konnte innerhalb der gewählten Timeoutzeit nicht ausgeführt werden Dies könnte folgende Ursachen haben: - Gerät ist nicht mit der SPS Verbunden - Kommunikationsparameter fehlerhaft - CAN-Bus Teilnehmer nicht vorhanden
W#16#0002	Interner Bausteinfehler	Interner Bausteinfehler
W#16#0003	Keine oder mehr als eine Bausteinaktion angewählt	Es kann immer nur eine Bausteinfunktion gleichzeitig ausgeführt werden
W#16#0004	Empfangendes Leseergebnis > Reading Result Array	Das empfangene Leseergebnis ist Länger als 200 Bytes. Zum Empfangen von längeren Leseergebnissen siehe Kapitel 4.6
W#16#0005	100 < FreeCommand. iCommandLength <=0	Ungültige Länge des freien Kommandos Gültiger Wertebereich: [1...100]
W#16#0006	Antwort des freien Kommandos > 100 Byte	Die Antwort auf das gesendete freie Kommando ist länger 100 Byte.
W#16#0007	63 < CAN_ID < 0	Ungültige CAN-ID Gültiger Wertebereich: [0..63]
W#16#0008	Reserviert	Reserviert
W#16#0009	Kommunikationsfehler	Kommunikation zum Gerät kann nicht hergestellt werden. Dies könnte folgende Ursachen haben: - Ungültiger ID Parameter - Verbindung wurde nicht aufgebaut - Es wurde ein Telegramm > arrRecord empfangen
W#16#XX0A	Gerätefehler	Es ist ein Gerätefehler aufgetreten ('sFA XX') XX = Gerätefehler (siehe Gerätedokumentation)
W#16#000B	Ungültige Kommandoantwort	Die gewählte Aktion wurde nicht ausgeführt. Dies kann je nach Aktion die folgenden Ursachen haben: - Triggereinstellung in der SOPAS Gerätekonfiguration fehlerhaft - Gerät befindet sich nicht im „Run-Mode“
W#16#000C – W#16#000F	Reserviert	Reserviert

Fehlercode	Kurzbeschreibung	Beschreibung
W#16#0010	9 < Matchcode.nMatchNumber <= 0	Ungültige Matchcode Nummer. Gültiger Wertebereich: [1..9]
W#16#0011	75 < Matchcode.iLength <= 0	Ungültige Matchcode Länge. Gültiger Wertebereich: [1..75]
W#16#XX12	Matchcode / Save Permanent wurde nicht ausgeführt.	Die ausgeführte Aktion wurde nicht ausgeführt. Das Gerät wurde wieder in den „RUN-Mode“ versetzt. XX Fehlercode des aufgetretenen Fehlers.
W#16#0013	Wechsel in „RUN-Mode“ nicht möglich.	Das Gerät kann nicht zurück in den „RUN-Mode“ versetzt werden. Dies könnte folgende Ursache haben: - Kommandoverarbeitung dauert länger als die eingestellte Timeout-Zeit am FB.

Tabelle 5: Fehlercode

7 Beispiel

Abbildung 14 zeigt eine Beispielbeschaltung des SICK LECTOR CLV TCP CP FBs. Die TCP Verbindung zum SICK Sensor wurde im Vorfeld mit NetPro projektiert und die Verbindungsparameter (ID) und (LADDR) dem Funktionsbaustein übergeben (siehe Kapitel 3.2). Da der angeschlossene Sensor sich nicht in einem CAN-Netzwerk befindet, wird als CAN-ID fest eine null eingetragen.

Für das Triggerergebnis (im SOPAS Output-Format definiert) sind 200 Bytes im DB72 reserviert. Sollte das Leseergebnis länger sein, wird dies durch einen Fehler am Baustein signalisiert.

Programmaufruf:

```

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
    
```

Abbildung 14: Beispielbeschaltung des SICK LECTOR CLV TCP CP FBs

7.1 Matchcode ändern/anlegen

Um eine neue Matchcode Evaluationsbedingung anzulegen bzw. eine existierende zu ändern müssen zunächst die erforderlichen Parameterwerte angegeben werden.

- nMatchNumber: Matchcode Nummer (hier: Match5)
- nCodeType: Code Typ ("*" = Alle Code Typen)
- nLength: Länge des Codes
- iContentLength: Codeinhalt

```
// ===== 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.DBB	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	' '

Abbildung 15: Matchcode Parameter

Die Matchcode Aktion (bMatchcode) wird ausgeführt, sobald das Bit „bRequest“ mit einer positiven Flanke angetriggert wird.

```
// SICK Lector / CLV Function Block Example
```

M/W	16	"iCanID"	DEC	0
M	10.0	"bRequest"	BOOL	<input checked="" type="checkbox"/> true
M	10.2	"bReqDone"	BOOL	<input checked="" type="checkbox"/> true
M	10.3	"bReqBusy"	BOOL	<input type="checkbox"/> false
M	10.4	"bError"	BOOL	<input type="checkbox"/> false
M/W	14	"nErrorcode"	HEX	VW#16#0000

```
// Selection of the FB action to be execute
```

M	12.1	"bTriggerOn"	BOOL	<input type="checkbox"/> false
M	12.2	"bTriggerOff"	BOOL	<input type="checkbox"/> false
M	12.3	"bMatchCode"	BOOL	<input checked="" type="checkbox"/> true
M	12.4	"bComTest"	BOOL	<input type="checkbox"/> false
M	12.5	"bSavePermanent"	BOOL	<input type="checkbox"/> false
M	12.6	"bFreeCommand"	BOOL	<input type="checkbox"/> false

Abbildung 16: Matchcode Aktion starten

Die Matchcodeaktion ist abgeschlossen sobald das Bit „bReqDone = TRUE“ signalisiert. Dieses Beispiel legt auf dem Gerät die folgende Evaluationsbedingung an.

Abbildung 17: Angelegte Evaluationsbedingung

7.2 Triggersignal senden

Damit eine Triggerung über die SPS erfolgen kann, muss die Triggerquelle zuvor mit SOPAS-ET auf „Kommando“ eingestellt werden.

In diesem Beispiel kann das Lesetor über den FB geöffnet und geschlossen werden. Optional wird das Lesetor automatisch bei einem „Good Read“ geschlossen.

Start/Stop of Object Trigger



Trigger delay

A. Start by a. Start delay ms

B. Stop by or or b. Stop delay ms

Trigger Distribution

Distribute on

Abbildung 18: SOPAS Objekt-Triggereinstellungen

Die Aktion wird ausgeführt, sobald das Bit „bRequest“ mit einer positiven Flanke angetriggert wird. Das Lesetor ist geöffnet, wenn der Funktionsbaustein „bReqDone = TRUE“ signalisiert.

```
// SICK Lector / CLV Function Block Example
```

M/W	16	"iCanID"	DEC	0
M	10.0	"bRequest"	BOOL	<input checked="" type="checkbox"/> true
M	10.2	"bReqDone"	BOOL	<input checked="" type="checkbox"/> true
M	10.3	"bReqBusy"	BOOL	<input type="checkbox"/> false
M	10.4	"bError"	BOOL	<input type="checkbox"/> false
M/W	14	"nErrorcode"	HEX	VW#16#0000

```
// Selection of the FB action to be execute
```

M	12.1	"bTriggerOn"	BOOL	<input checked="" type="checkbox"/> true
M	12.2	"bTriggerOff"	BOOL	<input type="checkbox"/> false
M	12.3	"bMatchCode"	BOOL	<input type="checkbox"/> false
M	12.4	"bComTest"	BOOL	<input type="checkbox"/> false
M	12.5	"bSavePermanent"	BOOL	<input type="checkbox"/> false
M	12.6	"bFreeCommand"	BOOL	<input type="checkbox"/> false

Abbildung 19: Lesetor öffnen

Wenn der Code erfolgreich gelesen wurde „Good Read“ schließt das Gerät automatisch das Lesetor und sendet den gelesenen Code an die SPS. Der Funktionsbaustein speichert den gelesenen Code im Array „ReadingResult.arrResult“ des Nutzdatenbausteins (DB72). Der Ausgangsparameter RD_DONE zeigt für einen SPS Zyklus an, dass neue Daten empfangen

wurden. Der Parameter ReadingResult.iLength gibt an, wie viele Bytes empfangen wurden, bzw. gültig sind.

```
// ===== 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	'I'
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	'L'
DB72.DBB	306	"SICK LECTOR CLV6XX DATA".ReadingResult.arrResult[17]	CHARACTER	''

Abbildung 20: Leseergebnis