

SICK **Lector6xx / CLV6xx Funktionsbaustein**

Bausteinversion V1.X

SICK Lector6xx / CLV6xx PNDP
Funktionsbaustein für Siemens S7-1200 /
S7-1500 Steuerungen (TIA-Portal)



Versionshistorie

Version	Datum	Beschreibung
V1.0	21.02.2014	Initiale Version
V1.1	07.08.2014	Manuelle Flankenwertung (R_TRIG Bausteinaufrufe entfernt)

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 Unterstützte Feldbus Gateways / Sensoren	5
3.3 Konfiguration im TIA-Portal	5
4 Bausteinbeschreibung	7
4.1 Bausteinspezifikationen	7
4.2 Arbeitsweise	8
4.3 Verhalten im Fehlerfall	9
4.4 Timing	9
4.5 Werteübergabe	10
4.5.1 Matchcode	10
4.6 Free Command	13
4.6.1 Reading Result	13
4.7 Empfangen von Leseergebnissen > 200 Byte	15
5 Parameter	16
6 Fehlercodes	19
7 Beispiele	23
7.1 Matchcode ändern/anlegen	24
7.2 Gerät triggern / Empfang von Leseergebnissen	25

1 Zu diesem Dokument

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

1.1 Funktion dieses Dokuments

Diese Anleitung beschreibt den Umgang mit dem SICK_Lector_CLV6xx_PNDP 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_CLV6xx_PNDP wird zur Kommunikation zwischen einer Siemens S7-1200 / S7-1500 Steuerung und einem SICK Lector6xx / CLV6xx Codeleser verwendet.

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

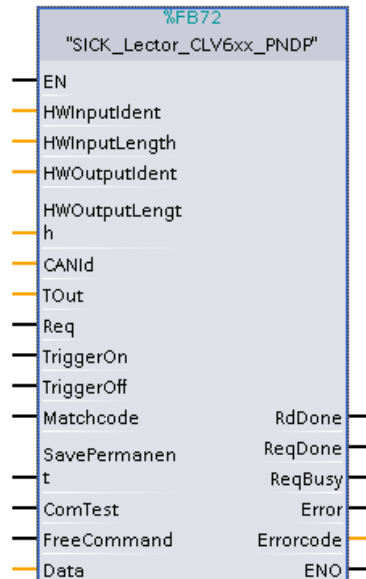


Abbildung 1: Darstellung des Funktionsbausteins in FUP

Funktionalität des Funktionsbausteins:

- Senden eines Software -Triggers über die SPS
- Empfang von geräteseitig gesendeten Telegrammen, die im SOPASⁱ Ausgabeformat konfiguriert werden können (Leseergebnisse)
- Anlegen / Ändern einer Evaluationsbedingung für einen Matchcode
- Permanentes Speichern aller Geräteparameter im Gerät
- Ausführen eines Kommunikationstests
- Kommunikation über frei wählbare Gerätekommandos (SICK CoLaⁱⁱ Protokoll)
- Ansprechen von Geräten in einem SICK CAN-Bus Netzwerk

ⁱ SOPAS ist ein Engineering Tool zum parametrieren von SICK Sensoren.

ⁱⁱ Die command language (CoLa) ist ein SICK internes Protokoll zur Kommunikation mit SOPAS Geräten.

3 Hardwarekonfiguration

3.1 Unterstützte SPS-Steuerungen

Der Funktionsbaustein kann nur mit Simatic S7-Steuerungen der 1200er oder 1500er Familie betrieben werden. Es werden nur Steuerungen unterstützt, die die verwendete Feldbus-schnittstelle direkt integriert haben. Ein Datenaustausch über einen Kommunikationsprozessor (CP- Baugruppe) wird nicht unterstützt.

3.2 Unterstützte Feldbus Gateways / Sensoren

Der SICK Sensor kommuniziert über einen Feldbus (Profibus/Profinet) mit der Steuerung. Sollte der Sensor die oben genannten Feldbusse nicht direkt unterstützen, können Gateway-Module eingesetzt werden.

Folgende Gateways werden vom Funktionsbaustein unterstützt:

- CDM 425 (Profinet), ab Firmware Version V3.31
- CDF 600-2 (Profibus und Profinet)
- CDF 600 (Profibus), ab Firmware Version V1.15
- CDM 420 inkl. CMF400 Profibus Modul, ab Firmware Version V1.100

3.3 Konfiguration im TIA-Portal

Bevor der Funktionsbaustein verwendet werden kann, muss in der Hardwarekonfiguration des TIA-Portals der entsprechende Sensor bzw. das entsprechende Gateway projiziert werden. Im ersten Schritt muss die entsprechende Gerätestammdatei (GSD / GSDML) in die Hardwarebibliothek importieren werden.

Der Funktionsbaustein ist speziell für den Handshake Modus (HS) ausgelegt. Bitte nur Module aus der Kategorie „Handshake (HS)“ verwenden, die mit einer Länge zwischen 8...128 Bytes definiert sind. Die verwendeten Adressen dürfen im Peripheriebereich oder außerhalb projiziert werden. Eine Adresszuweisung auf Peripheriebereiche, denen ein Teilprozessab-bild mit OB6x-Anbindung (Taktsynchronalarml) zugeordnet ist, darf nicht verwendet werden, da in diesem Fall eine konsistente Datenübertragung nicht mehr gewährleistet werden kann.

Abbildung 2 zeigt eine Beispielkonfiguration des SICK CLV6xx Barcodelesers. Die für den Funktionsbaustein benötigten Hardware-Kennungen stehen in den Eigenschaften der einzelnen Module.

The screenshot displays the SICK configuration software interface. At the top, the breadcrumb path is: ...3-1 PN] > Distributed I/O > PROFINET IO-System (100): PN/IE_1 > CLV6xx. The main window shows a 3D model of the CLV6xx barcode reader. Below the model is the 'Device overview' table:

Module	Rack	Slot	I address	Q addr...	Type
CLV6xx	0	0			CLV6xx HandShak...
Interface	0	0 X1			CLV6xx
Ctrl Bits in_1	0	1	0...1		Ctrl Bits in
Ctrl Bits out_1	0	2		0...1	Ctrl Bits out
32 Byte Input (HS)_1	0	3	2...33		32 Byte Input (HS)
32 Byte Output (HS)_1	0	4		2...33	32 Byte Output (HS)
	0	5			

The 'Properties' window for the '32 Byte Input (HS)_1' module is open, showing the 'General' tab. The 'Hardware identifier' field is highlighted with a blue box and a blue arrow pointing to the value '268'.

The 'Hardware catalog' on the right side of the interface shows the following structure:

- Filter
- Head module
- Module
 - Data Input Modules (HS)
 - 8 Byte Input (HS)
 - 12 Byte Input (HS)
 - 16 Byte Input (HS)
 - 20 Byte Input (HS)
 - 32 Byte Input (HS)
 - 64 Byte Input (HS)
 - 128 Byte Input (HS)
 - Ctrl Bits in
 - Ctrl Bits out
 - Data Output Modules (HS)
 - 8 Byte Output (HS)
 - 12 Byte Output (HS)
 - 16 Byte Output (HS)
 - 20 Byte Output (HS)
 - 32 Byte Output (HS)
 - 64 Byte Output (HS)
 - 128 Byte Output (HS)
 - Parameter Modules

Abbildung 2: Hardwarekonfiguration

Die Größe der In-/Out Module gibt an, wie viele Daten in einem Feldbuszyklus ausgetauscht werden können. Sollte ein Telegramm länger als das projektierte Modul sein, werden die Daten über mehrere SPS-Zyklen fragmentiert übertragen (Handshaking).

4 Bausteinbeschreibung

Der Funktionsbaustein SICK_Lector_CLV6xx_PNDP vereinfacht die Benutzung des Lector6xx / CLV6xx Codelesers an S7-1200 / S7-1500 Steuerungen. Der Baustein ermöglicht den Datenaustausch über eine in der Hardwarekonfiguration projektierten Profibus/Profinet Verbindung.

Der Baustein fragmentiert die Daten automatisch, sobald diese nicht in einem Feldbus-Zyklus übertragen / empfangen werden können.

Der Funktionsbaustein ist ein asynchron arbeitender FB d.h. die Bearbeitung erstreckt sich über mehrere FB-Aufrufe. Der Funktionsbaustein muss hierfür zyklisch im Anwenderprogramm aufgerufen werden.

Der Baustein kapselt den Funktionsbaustein SICK_CCOM_PNDP (FB10), der die Kommunikation zwischen SPS und Sensor ermöglicht. Die Funktionen SICK_GetValue / SICK_SetValue werden intern zur Interpretation / Erstellung der Geräte-Telegramme verwendet.

4.1 Bausteinspezifikationen

Bausteinname:	SICK_Lector_CLV6xx_PNDP
Bausteinnummer:	FB72
Version:	1.1
Unterstützte Steuerungen:	S7-1200 S7-1500
Verwendete Bausteine:	DPRD_DAT DPWR_DAT MOVE_BLK TON_TIME SICK_CCOM_PNDP SICK_GetValue SICK_SetValue
Verwendete PLC-Datentypen:	ST_SICK_Lector_CLV6xx
Optimierter Bausteinzugriff	Ja
Bausteinaufruf:	Zyklisch
Verwendete globale Variablen:	keine
Erstelsprache:	S7-SCL

4.2 Arbeitsweise

Um den Lector6xx / CLV6xxx Baustein einsetzen zu können, müssen zunächst die folgenden Kommunikationsparameter angegeben werden:

#HWInputIdent: Hardware-Kennung des projektierten Input-Moduls. Die Kennung wird bei der Hardwareprojektierung vom TIA-Portal festgelegt (siehe Abbildung 2).

#HWInputLength: Bytelänge des projektierten Input-Moduls (siehe Abbildung 2).

#HWOutputIdent: Hardware-Kennung des projektierten Output-Moduls. Die Kennung wird bei der Hardwareprojektierung vom TIA-Portal festgelegt (siehe Abbildung 2).

#HWOutputLength: Bytelänge des projektierten Output-Moduls (siehe Abbildung 2).

#Data: Der Funktionsbaustein benötigt eine Instanz vom PLC-Datentyp ST_SICK_Lector_CLV6xx. Dieser Datentyp beschreibt Ein- und Ausgabeparameter für die einzelnen Bausteinaktionen. Um den Datentyp nutzen zu können, muss in einem Datenbaustein eine Variable vom Typ ST_SICK_Lector_CLV6xx angelegt werden. Diese Variable muss anschließend dem Funktionsbaustein übergeben werden.

Um eine Bausteinaktion (#TriggerOn, #Matchcode, 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 #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 #ReqBusy signalisiert.

Wenn der Baustein am Ausgangsparameter #ReqDone = TRUE signalisiert, wurde die Aktion erfolgreich durchgeführt. Wurden bei dieser Aktion (z.B. #FreeCommand) Daten vom Gerät angefordert, werden diese in der jeweiligen Datenstruktur des Datentyps ST_SICK_Lector_CLV6xx (#Data) kopiert.

Daten die per Triggerbefehl (#TriggerOn, #TriggerOff) oder direkt vom Gerät gesendet werden (z.B. direkter Trigger über eine Lichtschranke), werden in der Datenstruktur (ReadingResult.arrResult) abgelegt. Der Ausgangsparameter #RdDone 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 Abbildung 7).

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 Parameter (#Error, Errorcode) des FBs behalten solange ihren Wert, bis ein neuer Auftrag gestartet wird.

4.4 Timing

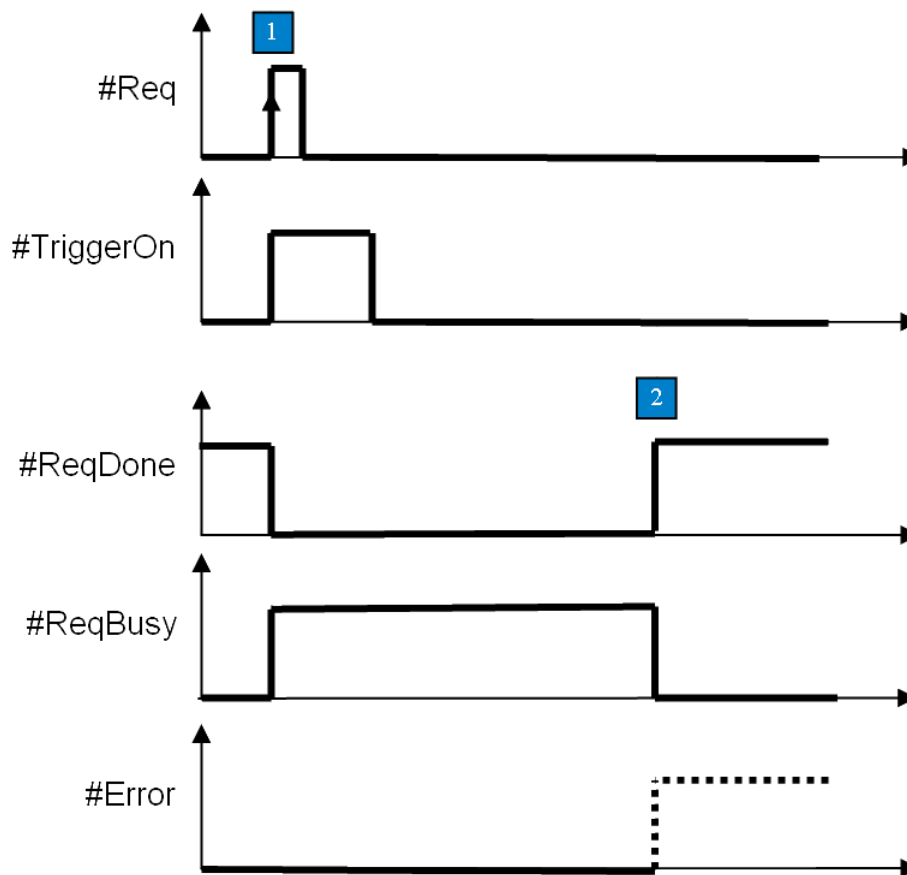


Abbildung 3: Timing Diagramm

1: Anforderung durch Pos Flanke an #Req. Die gewünschte Aktion (hier #TriggerOn) 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 #ReqDone beendet. Wenn die Aktion fehlerhaft verläuft, wird mit #Error beendet. Bei Abbruch mit #Error enthält der Parameter #Errorcode den aufgetretenen Fehlercode.

4.5 Werteübergabe

Der zum Funktionsbaustein zugehörige PLC-Datentyp ST_SICK_Lector_CLV6xx beinhaltet Ein- und Ausgabeparameter der unterstützten Bausteinaktionen. Die Datenstruktur ist fest vordefiniert und darf, bis auf den Eintrag „ReadingResult.arrResult“, nicht geändert werden (siehe Kapitel 4.7: Empfangen von Leseergebnissen > 200 Byte).

ST_SICK_Lector_CLV6xx							
	Name	Data type	Default value	Accessible ...	Visible in ...	Setpoint	Comment
1	Matchcode	Struct		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	==Matchcode==
2	sName	String[10]	"	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Matchcode number (Match[1..9]) (Input)
3	nCodeType	Char	"	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Code type see device documentation. (Example: 'd'= EAN-Code; 's'=QR-...
4	iMinMaxLength	USInt	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Sets the min and may length. 0= Don't care (Input)
5	sContent	String[75]	"	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Matchcode content
6	FreeCommand	Struct		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	==Free Command==
7	sCommand	String[100]	"	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Command (SICK CoLa-A protocol without [STX]/[ETX] framing) (In)
8	sResult	String[100]	"	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Result (SICK CoLa-A protocol) (Out)
9	ReadingResult	Struct		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	==Reading Result==
10	iCounter	USInt	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	This counter is incremented if a new reading result has arrived (In)
11	iLength	Int	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	byte length of the reading result (Out)
12	arrResult	Array[1..200] of Byte		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Reading result data defined in the SOPAS output format (Out)

Abbildung 4: ST_SICK_Lector_CLV6xx PLC-Datentyp

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 Matchcode Aktion ausgeführt wird, müssen in der Struktur Matchcode die folgenden Parameter angegeben werden. Die folgende Abbildung zeigt die PLC-Parameter im Vergleich zur Parametrieroberfläche im SOPAS Engineeringtool.

Abbildung 5: Evaluation Condition

Parameter	Deklaration	Datentyp	Beschreibung
Matchcode. sName	Input	String[10]	<p>Name des Matchcodes.</p> <p>Der Name darf keine Sonderzeichen und Leerzeichen enthalten, sowie nicht mit einer Ziffer beginnen.</p> <p>Gültige Zeichen: [a..z], [A..Z], [0..9]</p>
Matchcode. nCodeType	Input	Char	<p>Gewünschter Code Typ, auf der sich die Evaluationsbedingung beziehen soll.</p> <p>'a' = Codabar 'b' = Code39 'c' = UPC 'd' = EAN 'e' = Interleaved25 'f' = C25IND 'g' = MSI Code 'h' = Code93 'i' = Code128 'j' = MC Codabar 'm' = MC Codabar 'n' = EAN128 'o' = Pharma 'p' = PostNet 'q' = C25INDB 'r' = RSS Code 's' = QR Code 't' = Code49 'u' = Micro PDF417 'v' = PDF417 'w' = Datamatrix 'x' = Auxiliary 'y' = MC Zellweger Barcode 'z' = Zellweger Barcode 'A' = ADDON Code 'J' = Japanese Postal 'P' = Auto Setup 'Q' = Codablock F 'R' = Reflector polling 'S' = Auxiliary 2D 'T' = ADDON 2D 'U' = Auto Setup 2D 'V' = Reflector Polling 2D 'X' = Maxicode 'Z' = Aztec code '*' = Don't care</p>

Parameter	Deklaration	Datentyp	Beschreibung
Matchcode. iMinMaxLength	Input	USInt	Minimale und maximale Codelänge. 0 = Beliebige Codelänge Gültiger Wertebereich: [0..255]
Matchcode. sContent	Input	String[75]	Matchcodeinhalt

Tabelle 1: Matchcode Parameter

4.6 Free Command

Mit Hilfe des freien Kommandos hat man die Möglichkeit über ein gültiges Geräte-Kommando (CoLa-Protokoll) mit dem SICK Sensor zu kommunizieren. Hierfür ist es erforderlich, das Kommando in dem String „sCommand“ der Struktur „FreeCommand“ zu hinterlegen. Die Kommandos können der Gerätebeschreibung oder dem SOPAS Engineeringtool entnommen werden.

Parameter	Deklaration	Datentyp	Beschreibung
FreeCommand.sCommand	Input	String [100]	Frei wählbares CoLa Kommando (Kommandos siehe Gerätdokumentation).
FreeCommand.sResult	Output	String [100]	Empfangende Antwort des gesendeten CoLa Telegramms.

Tabelle 2: Free Command Parameter

4.6.1 Reading Result

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

Parameter	Deklaration	Datentyp	Beschreibung
ReadingResult.iCounter	Output	USInt	Der Empfangszähler wird um eins inkrementiert, sobald ein neues Leseergebnis empfangen wurde. Wertebereich: [0..255]
ReadingResult.iLength	Output	UInt	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 empfangenen Daten beträgt 200 Bytes. Kapitel 4.7 beschreibt das Vorgehen beim Empfang von längeren Datentelegrammen.

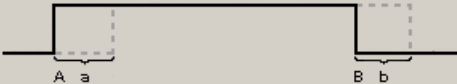
Tabelle 3: Reading Result Parameter

In der Objekttriggersteuerung wird festgelegt, wann das Lesetor geöffnet bzw. geschlossen wird. Nach jedem Lesetor sendet der Sensor ein Leseergebnis zur SPS-Steuerung.

Um das angeschlossene Gerät über die Triggerfunktion des Funktionsbausteins zu triggern, muss die SOPAS Einstellungen unter dem Menüpunkt *Parameter* → *Reading Configuration* → *Objekt Trigger Control* so vorgenommen werden, dass das Triggerfenster über ein „Kommando“ geöffnet bzw. wenn nötig, wieder geschlossen wird.

- Start mit "SOPAS-Kommando" (#TrigerOn Kommando kann verwendet werden)
- Stop mit "SOPAS-Kommando" (#TriggerOff Kommando kann 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 1000ms).

Start/Stop of Object Trigger



Trigger delay

A. Start by a. Start delay ms


B. Stop by or or b. Stop delay ms


Reading gate length ms

Abbildung 6: Triggereinstellung (SOPAS)

Das Ausgabeformat definiert den Inhalt des Telegramms, das vom Gerät gesendet wird, sobald das Triggerfenster geschlossen wurde. Für das Ausgabeformat stehen verschiedene Konfigurationsmöglichkeiten zur Verfügung.


Output Format 1

 Wizard



If Good read

For each code

 var s

Else

NoRead

Abbildung 7: SOPAS Ausgabeformat

4.7 Empfangen von Leseergebnissen > 200 Byte

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

Änderung im PLC-Datentyp (ST_SICK_Lector_CLV6xx):

In dem mitgelieferten PLC-Datentyp muss die Arraygröße der Variable ReadingResult.arrResult verändert werden (bis maximal 500 Byte).

▼ ReadingResult	Struct		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	==Reading Result==
■ iCounter	USInt	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	This counter is incremented if a new reading result has arrived (In)
■ iLength	Int	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	byte length of the reading result (Out)
▶ arrResult	Array[1..200] of Byte		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Reading result data defined in the SOPAS output format (Out)

Abbildung 8: Änderung PLC-Datentyp

Änderung im Programmcode (SICK_Lector_CLV6xx_PNDP):

Im Programmcode des SICK_Lector_CLV6xx_PNDP Bausteins muss anschließend die neu definierten Arraygröße des Leseergebnisses eingetragen werden.

```

28  (*===== INITIALISATION =====*)
29  #iRecordSize:= 500;           (*Length of the arrRecord array*)
30  #iCommandSize:= 500;         (*Length of the arrCommand array*)
31  #iReadingResultSize:= 200;    (*Length of the reading result array*)

```

Abbildung 9: Änderung Funktionsbaustein

Nach den Änderungen müssen die geänderten Bausteine neu kompiliert und zur SPS übertragen werden.

5 Parameter

Parameter	Deklaration	Datentyp	Beschreibung
HWInputIdent	Input	HW_IO	Hardware-Kennung des projektierten Input Moduls (siehe Hardwarekonfiguration). In der Hardwarekonfiguration dürfen nur HS-Module (Handshake-Module) mit einer maximalen Länge von 128Byte verwendet werden.
HWInputLength	Input	USInt	Länge des verwendeten Input-Moduls in der Hardwarekonfiguration. Gültiger Wertebereich: [8..128]
HWOutputIdent	Input	HW_IO	Hardware-Kennung des projektierten Output Moduls (siehe Hardwarekonfiguration). In der Hardwarekonfiguration dürfen nur HS-Module (Handshake-Module) mit einer maximalen Länge von 128Byte verwendet werden.
HWOutputLength	Input	USInt	Länge des verwendeten Output-Moduls in der Hardwarekonfiguration. Gültiger Wertebereich: [8..128]
CANId	Input	USInt	CAN-ID des anzusprechenden Sensors. Wenn kein SICK 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	Zeit, nachdem ein Timeout-Fehler ausgelöst wird. Wenn dieser Parameter nicht beschaltet ist, beträgt die Timeout Zeit standardmäßig 10 Sekunden. Bitte beachten Sie, dass einige Kommandos eine längere Bearbeitungszeit benötigen (z.B. Speicherkommandos).
Req	Input	Bool	Positive Flanke: Ausführen der gewählten Bau-steinaktion.

Parameter	Deklaration	Datentyp	Beschreibung
TriggerOn	Input	Bool	<p>Bausteinaktion: Ausführen eines Geräte Triggers (Triggerfenster öffnen).</p> <p>Diese Aktion setzt voraus, dass die Objekttriggersteuerung (SOPAS) für das öffnen des Lesetors auf „Command“ eingestellt ist.</p> <p>Das vom Gerät gesendete Leseergebnis (definiert im SOPAS Ausgabeformat) wird in der Variablen „ReadingResult.arrResult“ abgelegt (PLC Datentyp: ST_Lector_CLV6xx).</p>
TriggerOff	Input	Bool	<p>Bausteinaktion: Ausführen eines Geräte Triggers (Triggerfenster schließen)</p> <p>Diese Aktion setzt voraus, dass die Objekttriggersteuerung (SOPAS) für das schließen des Lesetors auf „Command“ eingestellt ist.</p> <p>Das vom Gerät gesendete Leseergebnis (definiert im SOPAS Ausgabeformat) wird in der Variablen „ReadingResult.arrResult“ abgelegt (PLC Datentyp: ST_Lector_CLV6xx).</p>
Matchcode	Input	Bool	<p>Bausteinaktion: Evaluation Condition erstellen/Ändern.</p> <p>Die Aktion setzt voraus, dass die Parameter der Struktur „Matchcode“ (PLC Datentyp: ST_SICK_Lector_CLV6xx) mit gültigen Werten belegt ist (siehe Kapitel 4.5.1).</p>
SavePermanent	Input	Bool	<p>Bausteinaktion: Permanentes Speichern aller Geräteparameter im Gerät.</p> <p>Ist der Sensor mit einem CMC-Modul (Cloning-Modul) verbunden, kann die Bausteinaktion unter Umständen länger >10s dauern. In diesem Fall muss die Timeout Zeit (#TOut) angepasst werden.</p>
ComTest	Input	Bool	<p>Bausteinaktion: Ausführen eines Kommunikationstests.</p> <p>#ReqDone= TRUE: Kommunikation OK #ReqDone= FALSE: Kommunikation nicht OK</p>

Parameter	Deklaration	Datentyp	Beschreibung
FreeCommand	Input	Bool	<p>Bausteinaktion: Ausführen eines freien Kommandos (siehe Kapitel 4.6).</p> <p>Das zu übertragende Kommando wird in der Variablen „FreeCommand.sCommand“ definiert (PLC Datentyp: ST_SICK_Lector_CLV6xx).</p> <p>Die Kommandoantwort steht nach einer erfolgreichen Übertragung (#ReqDone= TRUE) im Result-String „FreeCommand.sResult) zur Verfügung.</p>
Data	Input/Output	ST_SICK_Lector_CLV6xx	<p>Übergabe der Variablen vom Typ ST_SICK_Lector_CLV6xx die für die Parametrierung der Bausteinfunktionen sowie für das Ablegen des Leseergebnisses benötigt wird.</p> <p>Diese Variable muss in einem Datenbaustein angelegt werden und wird dem Funktionsbaustein übergeben.</p>
RdDone	Output	Bool	<p>Positive Flanke: Neues Leseergebnis empfangen</p> <p>Das Leseergebnis, sowie die gültige Länge wird in der Struktur „ReadingResult“ (PLC Datentyp: ST_SICK_Lector_CLV6xx) abgelegt (siehe Kapitel 4.6.1).</p>
ReqDone	Output	Bool	<p>Zeigt an, ob die gewählte Bausteinaktion fehlerfrei durchgeführt wurde.</p> <p>TRUE: Bearbeitung abgeschlossen FALSE: Bearbeitung nicht abgeschlossen</p>
ReqBusy	Output	Bool	Bausteinaktion befindet sich in Bearbeitung.
Error	Output	Bool	<p>Fehler Bit:</p> <p>FALSE: Kein Fehler TRUE: Abbruch mit Fehler</p>
Errorcode	Output	DWord	Fehlerstatus (siehe Fehlercodes)

Tabelle 4: Bausteinparameter

6 Fehlercodes

Der Parameter #Errorcode enthält folgende Fehlerinformationen:

- Fehler des SICK_Lector_CLV6xx_PNDP Funktionsbausteins
- Fehler des SICK_CCOM_PNDP Funktionsbausteins
- Fehler der SICK_GetValue / SICK_SetValue Funktionen
- Fehler der Siemens Funktionen DPRD_DAT / DPWR_DAT
- Fehler die geräteseitig gesendet werden

Fehlercode	Kurzbeschreibung	Beschreibung
16#0000_0000	Kein Fehler	Kein Fehler
16#0000_0001	Timeout (SICK_CCOM_PNDP)	Der Auftrag konnte innerhalb der gewählten Timeoutzeit nicht ausgeführt werden. Dies könnte folgende Ursache haben: <ul style="list-style-type: none"> - Gerät ist nicht mit der SPS verbunden - Gerät sendet keine Kommandoantwort (Echo) - Verarbeitungszeit des Kommandos > Timeout Zeit - CAN-Bus Teilnehmer nicht vorhanden
16#0000_0002	Ungültige Modul Länge (Input)	Die in der Hardwarekonfiguration projektierte Länge des Input Moduls ist ungültig. Gültiger Modullänge: [8..128]
16#0000_0003	Ungültige Modul Länge (Output)	Die in der Hardwarekonfiguration projektierte Länge des Output Moduls ist ungültig. Gültiger Modullänge: [8..128]
16#XXXX_0004	DPWR_DAT Error	Fehler beim schreiben auf das angegebene Output Modul. Bitte prüfen Sie die Hardware-Kennung, sowie die Länge des Output-Moduls. XXXX: Fehlercode der Siemens Funktion „DPWR_DAT“ (siehe Informationssystem TIA-Portal).
16#XXXX_0005	DPRD_DAT Error	Fehler beim auslesen des angegebene Input Moduls. Bitte prüfen Sie die Hardware-Kennung, sowie die Länge des Input-Moduls. XXXX: Fehlercode der Siemens Funktion „DPRD_DAT“ (siehe Informationssystem TIA-Portal).
16#0000_0006 - 16#0000_0009	Kommunikationsfehler	Interne Kommunikationsfehler siehe Beschreibung des FB's SICK_CCOM_PNDP.

Fehlercode	Kurzbeschreibung	Beschreibung
16#XXXX_000A - 16#XXXX_000F	Reserviert	Reserviert
16#0000_0010	Timeout (SICK_CCOM_PNDP)	<p>Der Auftrag konnte innerhalb der gewählten Timeoutzeit nicht ausgeführt werden.</p> <p>Dies könnte folgende Ursache haben:</p> <ul style="list-style-type: none"> - Gerät ist nicht mit der SPS verbunden - Gerät sendet keine Kommandoantwort (Echo) - Verarbeitungszeit des Kommandos > Timeout Zeit - CAN-Bus Teilnehmer nicht vorhanden
16#XXXX_0011	Gerätefehler	<p>Es ist ein Gerätefehler aufgetreten.</p> <p>XXXX = Dieser Fehler wird vom angeschlossenen Gerät gesendet (siehe Gerätedokumentation)</p> <p>16#0001: Access denied 16#0002: Unknown index 16#0003: Unknown index 16#0004: Wrong condition 16#0005: Invalid data 16#0006: Unknown error 16#0007: Too many parameters 16#0008: Parameter missing 16#0009: Wrong Parameter 16#000A: No Write access 16#000B: Unknown command 16#000C: Unknown command 16#000D: Server busy 16#000E: Textstring too long 16#000F: Unknown event 16#0010: Too many parameter 16#0011: Invalid character 16#0012: No message 16#0013: No answer 16#0014: Internal error 16#0015: HubAddress: Wrong 16#0016: Hubaddress: Error 16#0017: Hubaddress: Error</p> <p>Für eine detaillierte Fehlerbeschreibung siehe Gerätebeschreibung.</p>

Fehlercode	Kurzbeschreibung	Beschreibung
16#XXXX_0012	SICK_GetValue Fehler	Fehler beim interpretieren des empfangenden Gerätekommandos (interner Bausteinfehler). XXXX = Fehlercode der Funktion SICK_GetValue (siehe Bausteinbeschreibung).
16#XXXX_0013	SICK_SetValue Fehler	Fehler beim erstellen eines Gerätekommandos (interner Bausteinfehler). XXXX = Fehlercode der Funktion SICK_SetValue (siehe Bausteinbeschreibung).
16#0000_0014	#CANId > 63	Ungültige CAN-ID Gültiger Wertebereich: [0..63]
16#0000_0015	Keine oder mehr als eine Bausteinaktion angewählt	Es kann immer nur eine Bausteinfunktion gleichzeitig ausgeführt werden.
16#0000_0016	Ungültige Kommandoantwort empfangen	Die gewählte Aktion wurde nicht ausgeführt, da die erwartete Geräteantwort nicht mit der gesendeten Antwort übereinstimmt. Dies kann je nach Aktion die folgenden Ursachen haben: - Triggereinstellung in der SOPAS Gerätekonfiguration fehlerhaft - Gerät befindet sich nicht im „Run-Mode“ - Ungültige Matchcode Argumente
16#XXXX_0017	Wechsel in „RUN-Mode“ nicht möglich.	Es konnte nicht sichergestellt werden, dass das Gerät zurück in den „RUN-Mode“ versetzt wurde. XXXX = Vorranggegangener Fehlercode (Word 0). Bitte überprüfen Sie den Status des angeschlossenen Gerätes.
16#XXXX_0018 - 16#XXXX_001F	Reserviert	Reserviert
16#0000_0020	FreeCommand.sCommand > arrCommand (500Byte)	Ungültige Länge des freien Kommandos Gültiger Wertebereich: [1...100]
16#0000_0021	Antwort des freien Kommandos > Result-String (FreeCommand.sResult [100Zeichen])	Die Antwort auf das gesendete freie Kommando ist länger 100 Zeichen.

Fehlercode	Kurzbeschreibung	Beschreibung
16#0000_0022	Stringlänge Matchcode.sName =0	Es wurde kein Matchcodename angegeben (Leer-String).
16#0000_0023	Matchcode.nCodeType ungültig	Die Eingabe des Matchcode Code Type ist fehlerhaft Gültiger Wertebereich: [16#20...16#7E]
#ReadingResult.iLength = -1	Leseergebnis > 200 Byte	Das empfangende Leseergebnis ist länger als 200 Byte. Zum empfangen von Leseergebnissen > 200 Byte siehe Kapitel 4.7.

Tabelle 5: Fehlercodes

7 Beispiele

Abbildung 10 zeigt eine Beispielbeschaltung des SICK_Lector_CLV6xx_PNDP Funktionsbausteins im OB1-Programm der Steuerung. In der Hardwarekonfiguration ist ein SICK CLV6xx Barcodeleser mit einer Prozessdatenbreite von 32 Byte Input (Hardware-Kennung: 268) und 32 Byte Output (Hardware-Kennung: 269) projektiert (siehe **Abbildung 2**). Da der CLV nicht zusammen mit weiteren Geräten in einem SICK CAN-Netzwerk betrieben wird, wird als CAN-ID fest eine null eingetragen.

Im DB1 (SICK_CodeReader_Data) wurde die Variable „CodeReader“ vom Typ ST_SICK_Lector_CLV6xx angelegt. Diese Struktur von Variablen beinhaltet Ein- und Ausgabeparameter der unterstützten Bausteinaktionen

Beispielprogramm:

```

1  (*SICK Lector / CLV6xx function block*)
2  ▢ "fbSICK_Lector_CLV6xx_PNDP" (HWInputIdent:= 268,
3    HWInputLength:= 32,
4    HWOutputIdent:= 269,
5    HWOutputLength:= 32,
6    Data:= "SICK_CodeReader_Data".CodeReader);

```

Abbildung 10: Programmcode (Beispiel)

SICK_CodeReader_Data							
	Name	Data type	Retain	Accessible ...	Visible in ...	Setpoint	Comment
1	▾ Static		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
2	▾ CodeReader	"ST_SICK_Lector_CLV6xx"	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	SICK Lector / CLV6xx
3	▸ Matchcode	Struct	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	==Matchcode==
4	▸ FreeCommand	Struct	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	==Free Command==
5	▸ ReadingResult	Struct	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	==Reading Result==

Abbildung 11: Datenbaustein (Beispiel)

7.1 Matchcode ändern/anlegen

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

Matchcode Name: 'MyMatch'
Code Typ: '*' (Alle Code Typen)
Code Länge: 12
Code Inhalt: 'Hello*'

"SICK_CodeReader_Data".CodeReader.Matchcode.sName	String	'MyMatch'
"SICK_CodeReader_Data".CodeReader.Matchcode.nCodeType	Character	'*'
"SICK_CodeReader_Data".CodeReader.Matchcode.iMinMaxLength	DEC	12
"SICK_CodeReader_Data".CodeReader.Matchcode.sContent	String	'Hello*'

Abbildung 12: Matchcode Parameter

Die Matchcode Aktion (#Matchcode) wird ausgeführt, sobald das Bit #Req mit einer positiven Flanke angetriggert wird.

"fbSICK_Lector_CLV6xx_PNDP".Req	Bool	<input checked="" type="checkbox"/> TRUE
"fbSICK_Lector_CLV6xx_PNDP".TriggerOn	Bool	<input type="checkbox"/> FALSE
"fbSICK_Lector_CLV6xx_PNDP".TriggerOff	Bool	<input type="checkbox"/> FALSE
"fbSICK_Lector_CLV6xx_PNDP".Matchcode	Bool	<input checked="" type="checkbox"/> TRUE
"fbSICK_Lector_CLV6xx_PNDP".SavePermanent	Bool	<input type="checkbox"/> FALSE
"fbSICK_Lector_CLV6xx_PNDP".ComTest	Bool	<input type="checkbox"/> FALSE
"fbSICK_Lector_CLV6xx_PNDP".FreeCommand	Bool	<input type="checkbox"/> FALSE
"fbSICK_Lector_CLV6xx_PNDP".ReqDone	Bool	<input checked="" type="checkbox"/> TRUE
"fbSICK_Lector_CLV6xx_PNDP".ReqBusy	Bool	<input type="checkbox"/> FALSE
"fbSICK_Lector_CLV6xx_PNDP".Error	Bool	<input type="checkbox"/> FALSE
"fbSICK_Lector_CLV6xx_PNDP".Errorcode	Hex	16#0000_0000

Abbildung 13: Starten der Bausteinfunktion (Matchcode)

Die Aktion ist abgeschlossen sobald das Bit #ReqDone = TRUE signalisiert. Die neu erstellte Matchcode Kondition kann anschließend mit Hilfe des SOPAS Engineeringtools eingesehen und ggf. geprüft oder verändert werden.

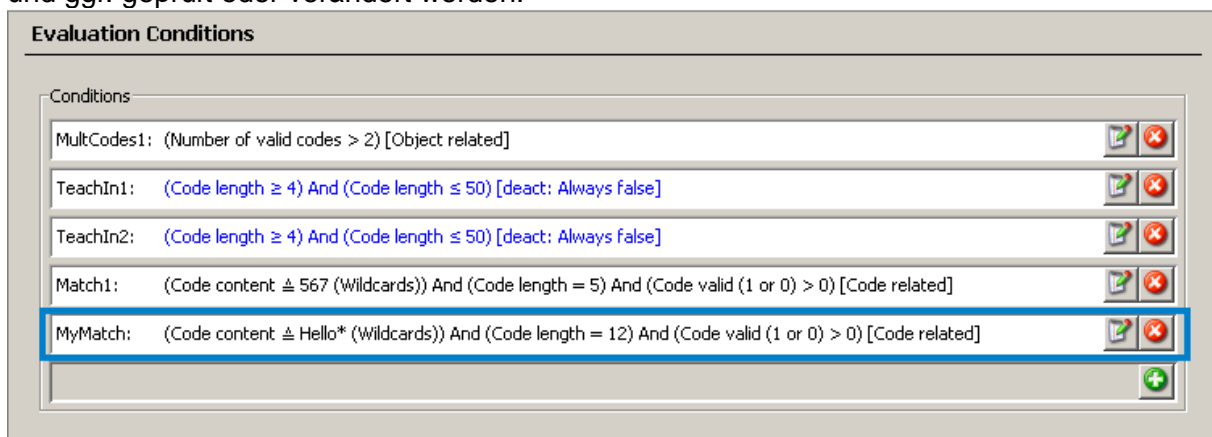


Abbildung 14: Überprüfung des erstellten Matchcodes in SOPAS

7.2 Gerät triggern / Empfang von Leseergebnissen

Damit eine Triggerung über den Funktionsbaustein erfolgen kann, muss die Triggerquelle zuvor mit SOPAS auf „Command“ 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.

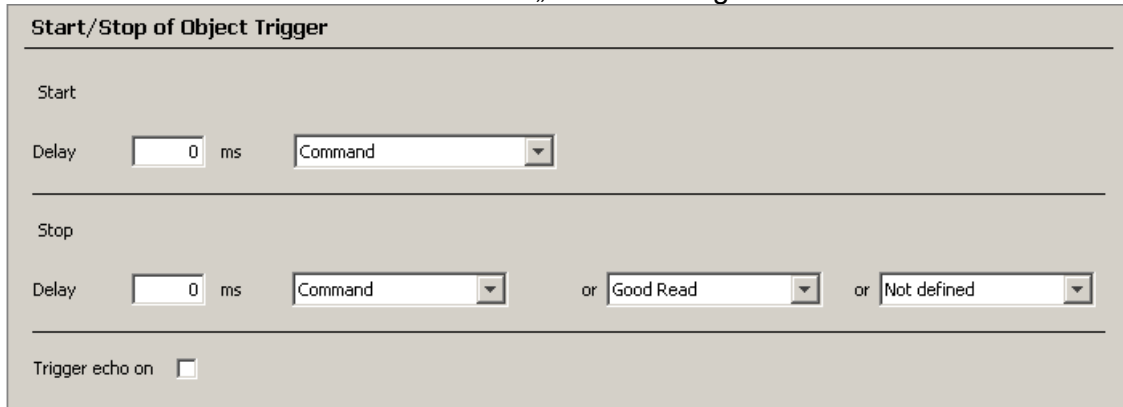


Abbildung 15: SOPAS Objekt-Triggereinstellungen

Die Aktion wird ausgeführt, sobald das Bit #Req mit einer positiven Flanke angetriggert wird. Das Lesetor ist geöffnet, wenn der Funktionsbaustein #ReqDone = TRUE signalisiert.

"fbSICK_Lector_CLV6xx_PNDP".Req	Bool	<input checked="" type="checkbox"/> TRUE
"fbSICK_Lector_CLV6xx_PNDP".TriggerOn	Bool	<input checked="" type="checkbox"/> TRUE
"fbSICK_Lector_CLV6xx_PNDP".TriggerOff	Bool	<input type="checkbox"/> FALSE
"fbSICK_Lector_CLV6xx_PNDP".Matchcode	Bool	<input type="checkbox"/> FALSE
"fbSICK_Lector_CLV6xx_PNDP".SavePermanent	Bool	<input type="checkbox"/> FALSE
"fbSICK_Lector_CLV6xx_PNDP".ComTest	Bool	<input type="checkbox"/> FALSE
"fbSICK_Lector_CLV6xx_PNDP".FreeCommand	Bool	<input type="checkbox"/> FALSE
"fbSICK_Lector_CLV6xx_PNDP".ReqDone	Bool	<input checked="" type="checkbox"/> TRUE
"fbSICK_Lector_CLV6xx_PNDP".ReqBusy	Bool	<input type="checkbox"/> FALSE
"fbSICK_Lector_CLV6xx_PNDP".Error	Bool	<input type="checkbox"/> FALSE
"fbSICK_Lector_CLV6xx_PNDP".Errorcode	Hex	16#0000_0000

Abbildung 16: Starten der Bausteinfunktion (TriggerOn)

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 Datenbausteins „CodeReader“. Der Ausgangsparameter #RdDone 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.

"SICK_CodeReader_Data".CodeReader.ReadingResult.iCounter	DEC	229
"SICK_CodeReader_Data".CodeReader.ReadingResult.iLength	DEC+/-	8
"SICK_CodeReader_Data".CodeReader.ReadingResult.arrResult[1]	Character	'T'
"SICK_CodeReader_Data".CodeReader.ReadingResult.arrResult[2]	Character	'e'
"SICK_CodeReader_Data".CodeReader.ReadingResult.arrResult[3]	Character	's'
"SICK_CodeReader_Data".CodeReader.ReadingResult.arrResult[4]	Character	't'
"SICK_CodeReader_Data".CodeReader.ReadingResult.arrResult[5]	Character	' '
"SICK_CodeReader_Data".CodeReader.ReadingResult.arrResult[6]	Character	'1'
"SICK_CodeReader_Data".CodeReader.ReadingResult.arrResult[7]	Character	'2'
"SICK_CodeReader_Data".CodeReader.ReadingResult.arrResult[8]	Character	'3'

Abbildung 17: Leseergebnis