

SICK **CoLa Kommunikationsbaustein für** **Autoident Geräte**

Bausteinversion: V1.X

SICK_CCOM_PNDP Funktionsbaustein für
Siemens S7-1200 / S7-1500 Steuerungen
(TIA-Portal)



Versionshistorie

Version	Datum	Beschreibung
V1.0	16.01.2014	Initiale Version
V1.1	07.08.2014	Manuelle Flankenbewertung (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
4.2.1 Empfangen von Leseergebnissen (Rd)	8
4.2.2 Gerätekommunikation über CoLa Kommandos (Req)	8
4.2.3 Timing	9
4.3 Verhalten im Fehlerfall	9
4.4 Rücksetzen der Kommunikation	9
4.5 Senden/Empfangen von Telegrammen >500Byte	10
4.6 Parameter	11
4.7 Fehlercodes	13
5 Beispiel	15
5.1 Senden/Empfangen von Telegrammen	16
5.2 Empfangen von Leseergebnissen	17

1 Zu diesem Dokument

Bitte lesen Sie dieses Kapitel sorgfältig, bevor Sie mit dieser Betriebsanleitung und dem SICK CoLa Kommunikationsbaustein arbeiten.

1.1 Funktion dieses Dokuments

Diese Betriebsanleitung beschreibt den Umgang mit dem SICK_CCOM_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_CCOM_PNDP unterstützt beim Profibus/Profinet Datenaustausch zwischen SICK Autolident Geräten und Siemens S7-1200 / S7-1500 Steuerungen.

Der Funktionsbaustein kann zur Kommunikation mit den folgenden SICK Sensoren verwendet werden:

- CLV6xx
- Lector6xx
- RFH6xx
- RFU6xx

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

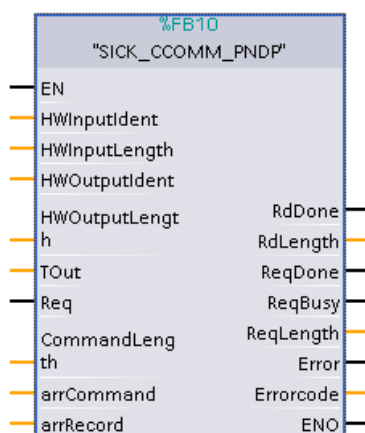


Abbildung 1: Darstellung des Funktionsbausteins in FUP

Funktionalität des Funktionsbausteins:

- Senden von CoLaⁱ Kommandos an ein SICK Sensor
- Empfangen von CoLa Antworten eines SICK Sensor
- Empfangen von geräteseitig gesendeten Telegrammen, die im SOPASⁱⁱ Ausgabeformat konfiguriert werden können (Leseergebnisse)

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

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

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. Der erste Schritt ist, die entsprechende Gerätestammdatei (GSD / GSDML) in die Hardwarebibliothek zu importieren.

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 Teilprozessabbild 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 eines SICK RFU6xx RFID Interrogator. Die für den Funktionsbaustein benötigten Hardware-Kennungen stehen in den Eigenschaften der einzelnen Module.

The screenshot shows the SIMATIC Manager HW Config interface. The main window displays a rack with modules. The 'Device overview' table lists modules including '20 Byte Input (HS)_1' and '8 Byte Output (HS)_1'. The 'Hardware catalog' on the right shows 'Data Input Modules (HS)' and 'Data Output Modules (HS)'. The 'Properties' window at the bottom shows the 'Hardware identifier' as 269.

Module	Rack	Slot	I address	Q addr...	Type	Order...	Firmware
RFU6xx	0	0			RFU6xx HandShak...		V1.0.0
Interface	0	0 X1			RFU6xx		
Ctrl Bits in_1	0	1	0...1		Ctrl Bits in		
Ctrl Bits out_1	0	2		0...1	Ctrl Bits out		
20 Byte Input (HS)_1	0	3	2...21		20 Byte Input (HS)		
8 Byte Output (HS)_1	0	4		2...9	8 Byte Output (HS)		
	0	5					

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_CCOM_PNDP vereinfacht die Benutzung von SICK Sensoren an S7-1200 / S7-1500 Steuerungen. Der Baustein ermöglicht das Senden und Empfangen von CoLa Telegrammen über eine in der Hardwarekonfiguration projektierten Profibus/Profinet Verbindung.

Der Baustein fragmentiert die Daten automatisch, sobald diese nicht in einem 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 SICK_CCOM_PNDP Baustein kapselt die Siemens Funktionsbausteine „DPRD_DAT“ und „DPWR_DAT“, die für den konsistenten Datenaustausch zwischen SPS und Sensor verwendet werden.

4.1 Bausteinspezifikationen

Bausteinname:	SICK_CCOM_PNDP
Bausteinnummer:	FB10
Version:	1.1
Unterstützte Steuerungen:	S7-1200 S7-1500
Verwendete Bausteine:	DPRD_DAT DPWR_DAT MOVE_BLK TON_TIME
Optimierter Bausteinzugriff	Ja
Bausteinaufruf:	Zyklisch
Verwendete globale Variablen:	keine
Erstelsprache:	S7-SCL

4.2 Arbeitsweise

Um den SICK_CCOM_PNDP Baustein einsetzen zu können, müssen zunächst die folgenden Parameter 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*).

#CommandLength: Zeichenlänge des zu übertragenden CoLa Kommandos.

#arrCommand: Datenbereich wo das zu sendende CoLa Kommando abgelegt wird. Der Datenbereich muss vom Programmierer selbst erstellt werden (Array[0...499] of Byte). Das Kommando muss ohne [STX] / [ETX] Rahmung angegeben werden.

#arrRecord: Datenbereich in den die vom Gerät gesendeten Telegramme abgelegt werden. Der Datenbereich muss vom Programmierer selbst erstellt werden (Array[0...499] of Byte).

4.2.1 Empfangen von Leseergebnissen (Rd)

Daten die geräteseitig gesendet werden (Rd), werden in das Record-Array #arrRecord geschrieben, sobald der Funktionsbaustein neue Daten empfangen hat. Das Bit #RdDone zeigt für einen SPS Zyklus den Empfang neuer Daten an. Die jeweilige Bytelänge des zuletzt empfangenden Telegramms kann dem Parameter #RdLength entnommen werden.

Eingangs- und Ausgangstelegramme werden vom Funktionsbaustein unabhängig behandelt. So ist es möglich Leseergebnisse zu empfangen, die geräteseitig z.B. über einen Hardware-Trigger, gesendet werden. Wird der Baustein ausschließlich zum empfangen von Leseergebnissen verwendet muss der Parameter #CommandLength mit dem Wert 0 beschaltet werden.

4.2.2 Gerätekommunikation über CoLa Kommandos (Req)

Bei der Kommunikation via CoLa Kommandos, wird das im Kommando-Array #arrCommand definierte Kommando zum Gerät übertragen. Die resultierende Antwort wird in den vom Parameter #arrRecord definierten Bereich abgelegt.

Sie starten die Übertragung, indem Sie den Parameter #Req mit einer positiven Flanke antriggern. Solange noch keine gültige Antwort auf das gesendete CoLa Kommando eingetroffen ist, wird dies über den Parameter #ReqBusy signalisiert. Sollte innerhalb der Timeout Zeit #TOut keine Antwort eingetroffen sein, wird die Bearbeitung mit einem Timeout Fehler #Errorcode abgebrochen. Der Ausgangsparameter #ReqDone zeigt an, dass eine Antwort auf das gesendete CoLa Kommando empfangen wurde (#ReqDone = TRUE). Eine neue Anfrage kann nur ausgeführt werden, wenn der Baustein nicht beschäftigt ist (#ReqBusy = FALSE).

4.2.3 Timing

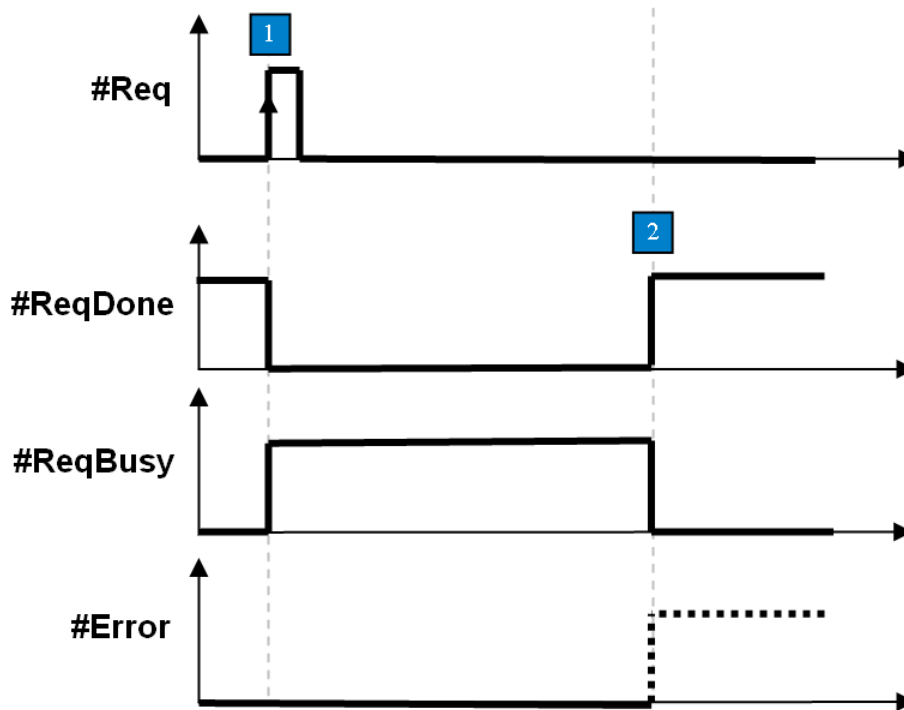


Abbildung 3: Timing Diagramm

1: Anforderung durch eine positive Flanke an #Req. Das vom Parameter #arrCommand referenzierte CoLa Kommando wird zum Sensor gesendet. Es kann immer nur ein Kommando zeitgleich gesendet werden.

2: Wenn das Kommando versendet ist und die Antwort empfangen wurden, wird die Aktion mit #ReqDone beendet. Wenn die Aktion fehlerhaft verläuft, wird mit #Error abgebrochen. Bei Abbruch mit #Error enthält #Errorcode den aufgetretenen Fehlercode.

4.3 Verhalten im Fehlerfall

Im Fehlerfall signalisiert das Errorbit #Error das ein Fehler aufgetreten ist. In diesem Fall wird über den Parameter #Errorcode ein Fehlercode ausgegeben. Das Errorbit #Error bleibt solange gesetzt, bis ein neuer Auftrag gestartet wird.

Wird der Baustein ausschließlich zum Empfang von Leseergebnissen verwendet (#CommandLength = 0), bleibt der Fehler #Error solange gesetzt, bis ein neues Leseergebnis empfangen wurde.

Sollte ein Leseergebnis empfangen werden, was länger als das Record-Array (#arrRecord) ist, wird die ausgelesene Länge #RdLength auf -1 gesetzt. Die Telegrammlänge bleibt solange gesetzt, bis ein neues Leseergebnis empfangen wurde.

4.4 Rücksetzen der Kommunikation

Das zurücksetzen der Kommunikation zwischen Gateway/Sensor und der PLC geschieht automatisch, sobald eine Differenz zwischen den im CM-Protokoll enthaltenen Zähler festgestellt wird.

4.5 Senden/Empfangen von Telegrammen >500Byte

Der Funktionsbaustein ist darauf ausgelegt, Telegramme bis zu einer Länge von 500 Bytes zu senden bzw. zu empfangen. Sollen längere Telegramme verarbeitet werden können, muss der Funktionsbaustein an den folgenden Stellen modifiziert werden:

Änderung in der Variablendeklaration:

Im InOut-Bereich der Bausteinschnittstelle müssen die Längen der folgenden Array-Variablen angepasst werden:

- #arrRecord (Zum empfangen von Telegrammen > 500 Byte)
- #arrCommand (Zum senden von Telegrammen > 500 Byte)

Interface							
	Name	Data type	Default value	Retain	Accessible ...	Visible in ...	Setpoint
15	InOut				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
16	arrCommand	Array[0..499] of Byte			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
17	arrRecord	Array[0..499] of Byte			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Abbildung 4: Telegramme > 500 Byte (Deklarationsänderungen)

Änderung im Programmcode des Funktionsbausteins:

Im Programmcode müssen die neu vergebenen Längen der Arrays angegeben werden.

- #iRecordSize (Größe des Record-Arrays #arrRecord)
- #iCommandSize (Größe des Kommando-Arrays #arrCommand)

```

55  (*===== INITIALISATION =====*)
56  #iRecordSize:= 500;  (*Length of the arrRecord array*)
57  #iCommandSize:= 500; (*Length of the arrCommand array*)
58

```

Abbildung 5: Telegramme > 500 Byte (Programmcodeänderungen)

4.6 Parameter

Parameter	Deklaration	Datentyp	Beschreibung
HWInputIdent	Input	HW_IO	Hardware-Kennung des projektierten Input Moduls (siehe Hardwarekonfiguration).
HWInputLength	Output	USInt	<p>Bytelänge des Input Moduls (siehe Hardwarekonfiguration)</p> <p>Gültiger Wertebereich: [8...128]</p>
HWOutputIdent	Input	HW_IO	Hardware-Kennung des projektierten Output Moduls (siehe Hardwarekonfiguration).
HWOutputLength	Output	USInt	<p>Bytelänge des Output Moduls (siehe Hardwarekonfiguration)</p> <p>Gültiger Wertebereich: [8...128]</p>
TOut	Input	Time	<p>Zeit, nachdem ein Timeout-Fehler ausgelöst wird.</p> <p>Wenn dieser Parameter nicht beschaltet ist, beträgt die Timeout Zeit standardmäßig 5 Sekunden.</p> <p>Bitte beachten Sie, dass einige CoLa Kommandos eine längere Bearbeitungszeit benötigen (z.B. Speicherkommandos).</p>
Req	Input	Bool	Positive Flanke: Sendet ein CoLa Kommando und wartet auf die entsprechende Antwort.
CommandLength	Input	UInt	Zeichenlänge des zu übertragenden CoLa Kommandos.
arrCommand	In/Out	Array[0..499] of Byte	<p>Datenbereich wo das zu sendende CoLa Kommando abgelegt ist.</p> <p>Das Kommando muss ohne [STX] / [ETX] Rahmung angegeben werden.</p>
arrRecord	In/Out	Array[0..499] of Byte	Datenbereich in den die vom Gerät gesendeten Telegramme abgelegt werden.
RdDone	Output	Bool	<p>Positive Flanke: Ein vom Gerät gesendetes Leseergebnis wurde empfangen (Formatierung siehe SOPAS Ausgabeformat).</p> <p>Sobald ein Leseergebnis empfangen wurde, ist das Bit für jeweils einen SPS-Zyklus gesetzt. Das Leseergebnis steht im Record-Array #arrRecord zur Verfügung.</p>

Parameter	Dekla- ration	Datentyp	Beschreibung
RdLength	Output	Int	Gibt die Bytelänge des empfangenden Le-seergebnisses an. Fehlerfall: Wird ein Telegramm empfangen, welches län-ger als das Record-Array (#arrRecord) ist, wird die Länge -1 ausgegeben.
ReqDone	Output	Bool	Zeigt an, ob ein CoLa Kommando abgeschickt und eine Antwort empfangen wurde. TRUE: Bearbeitung abgeschlossen FALSE: Bearbeitung noch nicht abgeschlossen Die Kommandoantwort steht im Record-Array #arrRecord zur Verfügung.
ReqBusy	Output	Bool	Auftrag ist in Bearbeitung.
ReqLength	Output	Int	Gibt die Bytelänge der Kommandoantwort an.
Error	Output	Bool	Fehler Status: 0: Kein Fehler 1: Abbruch mit Fehler
Errorcode	Output	DWord	Fehlerstatus (siehe Fehlercodes).

4.7 Fehlercodes

Der Parameter #Errorcode enthält die folgenden Fehlerinformationen:

Fehlercode	Kurzbeschreibung	Beschreibung
16#0000_0000	Kein Fehler	Kein Fehler
16#0000_0001	Timeout	<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 dem Gateway verbunden - Gerät sendet keine Kommandoantwort (Echo) - Verarbeitungszeit des Kommandos > Timeout Zeit
16#0000_0002	Ungültige Modul Länge (Input)	<p>Die in der Hardwarekonfiguration projektierte Länge des Input Moduls ist ungültig.</p> <p>Gültiger Modullänge: [8..128]</p>
16#0000_0003	Ungültige Modul Länge (Output)	<p>Die in der Hardwarekonfiguration projektierte Länge des Output Moduls ist ungültig.</p> <p>Gültiger Modullänge: [8..128]</p>
16#XXXX_0004	DPWR_DAT Error	<p>Fehler beim schreiben auf das angegebene Output Modul. Bitte prüfen Sie die Hardware-Kennung, sowie die Länge des Output-Moduls.</p> <p>XXXX: Fehlercode der Siemens Funktion „DPWR_DAT“ (siehe Informationssystem TIA-Portal).</p>
16#XXXX_0005	DPRD_DAT Error	<p>Fehler beim auslesen des angegebene Input Moduls. Bitte prüfen Sie die Hardware-Kennung, sowie die Länge des Input-Moduls.</p> <p>XXXX: Fehlercode der Siemens Funktion „DPRD_DAT“ (siehe Informationssystem TIA-Portal).</p>
16#0000_0006	Kommandoantwort > Record-Array	<p>Die empfangende Kommandoantwort passt nicht vollständig in das Record-Array (#arrRecord).</p> <p>Zum senden von CoLa-Telegrammen > 500 Byte siehe Kapitel 4.5</p>
16#0000_0007	Ungültiger Eingangsparameter (#CommandLength)	Zum senden eines Kommandos muss eine Kommandolänge (#CommandLength) >0 angegeben werden.
16#0000_0008	Ungültiger Eingangsparameter (#CommandLength)	Der Eingangsparameter (#CommandLength) ist größer als die Bytelänge des Kommando Arrays (#arrCommand).

Fehlercode	Kurzbeschreibung	Beschreibung
16#0000_0009	Fragmentierungsfehler	Interner Bausteinfehler
#RdLength = -1	Leseergebnis > Record-Array	Das empfangenden Leseergebnis passt nicht vollständig in das Record-Array (#arrRecord). Zum empfangen von Leseergebnissen > 500 Byte siehe Kapitel 4.5

5 Beispiel

Abbildung 6 zeigt eine Beispielbeschaltung des SICK_CCOM_PNDP Funktionsbausteins im OB1-Programm der Steuerung. In der Hardwarekonfiguration ist ein SICK Autolent Gerät mit einer Prozessdatenbreite von 20 Byte Input (Hardware-Kennung: 269) und 8 Byte Output (Hardware-Kennung: 270) projektiert (siehe *Abbildung 2*).

Sobald ein Leseergebnis bzw. eine Kommandoantwort empfangen wurde, wird diese in eine String-Variable im Datenbaustein „DeviceData“ abgespeichert. Ein Kommando kann mit Hilfe der Variablen-tabelle „Test“ zum Gerät gesendet werden. Die empfangenden Telegramme werden ebenfalls im String-Format dargestellt.

Beispielprogramm:

```

1  (*=====
2  Name:   Example program
3  Author: SICK AG
4  Date:   16.01.2014
5  =====
6  Description:
7  This example program calls up the SICK_CCOM_PNDP function block. A command can be sent using
8  the watch table "Test". The Data block "DeviceData" stores the incoming reading or command
9  result strings and counts it up in a counter variable.
10
11 The string "strCommand" in the data block defines the command, which is sent to the sensor.
12 =====)
13
14 (*Rising edge detection of the ReqDone flag*)
15 "fbR_TRIG"(CLK:="fbSICK_CCOMM_PNDP".ReqDone);
16
17 (*Incoming telegram is a reading result*)
18 IF "fbSICK_CCOMM_PNDP".RdDone THEN
19   Chars_To_Strg(Chars:= "DeviceData".ColaComm.arrRecord,           (*Store the result in a string*)
20                 pChars:= 0,
21                 Cnt:= INT_TO_UINT("fbSICK_CCOMM_PNDP".RdLength),
22                 Strg=> "DeviceData".ReadingResult.strResult);
23   "DeviceData".ReadingResult.iCounter:= "DeviceData".ReadingResult.iCounter+1; (*Increment counter value*)
24 END_IF;
25
26 (*Incoming telegram is a command result*)
27 IF "fbR_TRIG".Q THEN
28   Chars_To_Strg(Chars:= "DeviceData".ColaComm.arrRecord,           (*Store the result in a string*)
29                 pChars:= 0,
30                 Cnt:= INT_TO_UINT("fbSICK_CCOMM_PNDP".ReqLength),
31                 Strg=> "DeviceData".CommandResult.strResult);
32   "DeviceData".CommandResult.iCounter:= "DeviceData".CommandResult.iCounter+1; (*Increment counter value*)
33 END_IF;
34
35 (*Copy command which is defined in the strCommand variable*)
36 Strg_To_Chars(Strg:= "DeviceData".strCommand,
37               pChars:= 0,
38               Cnt=> #iCnt,
39               Chars:= "DeviceData".ColaComm.arrCommand);
40 "fbSICK_CCOMM_PNDP".CommandLength:= #iCnt;           (*Set the command length*)
41
42 (*Communication with the SICK AutoIdent device*)
43 IF "fbSICK_CCOMM_PNDP"(HWInputIdent:= 269,
44                       HWInputLength:= 20,
45                       HWOutputIdent:= 270,
46                       HWOutputLength:= 8,
47                       arrCommand:= "DeviceData".ColaComm.arrCommand,
48                       arrRecord:= "DeviceData".ColaComm.arrRecord);

```

Abbildung 6: Programmcode (Beispiel)

DeviceData			
	Name	Data type	Start value
1	Static		
2	strCommand	String	"
3	ColaComm	Struct	
4	arrCommand	Array[0..499] of Byte	
5	arrRecord	Array[0..499] of Byte	
6	ReadingResult	Struct	
7	strResult	String	"
8	iCounter	UInt	0
9	CommandResult	Struct	
10	strResult	String	"
11	iCounter	UInt	0

Abbildung 7: Datenbaustein (Beispiel)

5.1 Senden/Empfangen von Telegrammen

Das CoLa Kommando (hier: 'sRIO') wird ausgeführt, sobald das Bit #Req mit einer positiven Flanke angesteuert wird. Das Beispielprogramm kopiert das Kommando #strCommand in das Kommando-Array #arrCommand welches dem Funktionsbaustein übergeben wird. Die Kommandolänge #CommandLength resultiert aus der Anzahl der Zeichen des Kommandos (,sRIO' = 4).

Name	...	Display format	Monitor value	Modify value
"fbSICK_CCOMM_PNDP".Req		Bool	TRUE	TRUE
"fbSICK_CCOMM_PNDP".ReqDone		Bool	TRUE	
"fbSICK_CCOMM_PNDP".ReqBusy		Bool	FALSE	
"fbSICK_CCOMM_PNDP".Error		Bool	FALSE	
"fbSICK_CCOMM_PNDP".Errorcode		Hex	16#0000_0000	
"DeviceData".strCommand		String	'sRIO'	'sRIO'
"DeviceData".CommandResult.iCounter		DEC	1	
"DeviceData".CommandResult.strResult		String	'sRA 0 7 RFU630E 10 V1.32-23.0...	

Abbildung 8: Senden/Empfangen von Telegrammen

Die Kommandoantwort auf das gesendete Kommando (hier: 'sRA 0 7 RFU630E...') steht im Record-Array zur Verfügung, sobald sich der Wert des Ausgangsbits #ReqDone von FALSE auf TRUE ändert (positive Flanke). Der Parameter #ReqLength gibt an, wie viele Bytes empfangen wurden bzw. gültig sind. Das Beispielprogramm kopiert anschließend die Kommandoantwort in die String-Variable #DeviceData.ReadingResult.strResult.

5.2 Empfangen von Leseergebnissen

Um ein Leseergebnis empfangen zu können, muss das Gerät zunächst mit SOPAS-ET parametrisiert werden.

In diesem Beispiel wird eine Triggerung über ein Kommando vorgenommen. Das Triggerfenster wird automatisch vom Gerät geschlossen, sobald ein Code gelesen wurde.

Start/Stop of Object Trigger

Trigger delay: Time controlled

A, Start by: SOPAS-Command a, Start delay: 0 ms

B, Stop by: SOPAS-Command or Good Read or Timer / Tracking b, Stop delay: 0 ms

Reading gate length: 5000 ms

Reading gate on Reading gate off

Abbildung 9: Objekt-Trigger Einstellung (SOPAS-ET)

Nachdem das Triggerfenster geschlossen wurde, sendet das Gerät ein Leseergebnis zur SPS. Der Inhalt des Leseergebnisses kann im Ausgabeformat (SOPAS-ET) konfiguriert werden. In diesem Beispiel wird der String 'Hello World' gesendet.

Output Format 1

Wizard

?

Hello SPC World

0x20

Abbildung 10: Ausgabeformat (SOPAS-ET)

Das CoLa Kommando zur Triggerung (hier: 'sMN mTCgateon') wird ausgeführt, sobald das Bit #Req mit einer positiven Flanke angesteuert wird. Das Beispielprogramm kopiert das Kommando #strCommand in das Kommando-Array #arrCommand welches dem Funktionsbaustein übergeben wird. Die Kommandolänge #CommandLength resultiert aus der Anzahl der Zeichen des Kommandos ('sMN mTCgateon' = 13).

Name	...	Display format	Monitor value	Modify value
"fbSICK_CCOMM_PNDP".Req		Bool	<input checked="" type="checkbox"/> TRUE	TRUE
"fbSICK_CCOMM_PNDP".ReqDone		Bool	<input checked="" type="checkbox"/> TRUE	
"fbSICK_CCOMM_PNDP".ReqBusy		Bool	<input type="checkbox"/> FALSE	
"fbSICK_CCOMM_PNDP".Error		Bool	<input type="checkbox"/> FALSE	
"fbSICK_CCOMM_PNDP".Errorcode		Hex	16#0000_0000	
"DeviceData".strCommand		String	'sMN mTCgateon'	'sMN mTCgate...
"DeviceData".CommandResult.iCounter		DEC	2	
"DeviceData".CommandResult.strResult		String	'sAN mTCgateon 1'	
"DeviceData".ReadingResult.iCounter		DEC	1	
"DeviceData".ReadingResult.strResult		String	'Hello World'	

Abbildung 11: Empfangen von Leseergebnissen

Die Kommandoantwort auf das gesendete Kommando (hier: sAN mTCgateon 1') steht im Record-Array zur Verfügung, sobald sich der Wert des Ausgangsbits #ReqDone von FALSE auf TRUE ändert (positive Flanke). Der Parameter #ReqLength gibt an, wie viele Bytes empfangen wurden bzw. gültig sind. Das Beispielprogramm kopiert anschließend die Kommandoantwort in die String-Variable #DeviceData.ReadingResult.strResult.

Leseergebnisse, werden in das Record-Array geschrieben, sobald der Funktionsbaustein neue Daten empfangen hat. Das Bit #RdDone signalisiert für einen SPS Zyklus den Empfang neuer Daten an (Signalwechsel von FALSE auf TRUE). Der Parameter #RdLength gibt an, wie viele Bytes empfangen wurden bzw. gültig sind. Das Beispielprogramm kopiert anschließend das Leseergebnis in die String-Variable #DeviceData.ReadingResult.strResult.