Technische Information

SICK
CoLa Kommunikationsbaustein

SICK CCOM PN CP Funktionsbaustein für
Siemens Step7 Steuerungen

SICK
Sensor Intelligence.

Inhaltsverzeichnis

1 Zu diesem Dokument

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

1.1 Funktion dieses Dokuments

Diese Betriebsanleitung beschreibt den Umgang mit dem SICK CCOM_PN_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 CCOM_PN_CP unterstützt beim Datenaustausch zwischen SICK Geräten und S7-Steuerungen. Der Baustein unterstützt eine Profinet-Anbindung über ein Simatic CP-Modul (Kommunikationsprozessor). Steuerungen mit integrierter Profinet Controller werden nicht unterstützt.

Mit diesem Baustein können die folgenden SICK Sensoren angesteuert werden:
- CLV6xx
- LECTOR62x
- RFH62x
- RFU63x
- MSC800

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

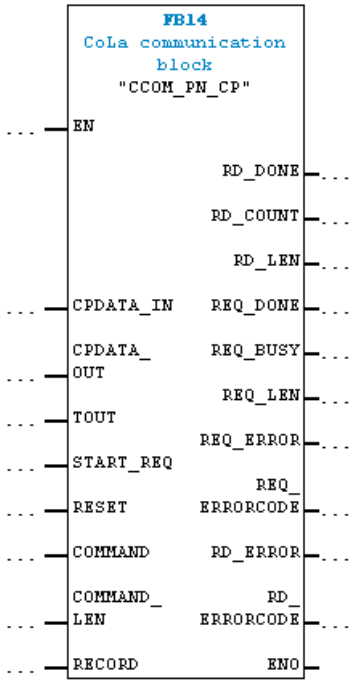

Abbildung 1: Darstellung des Funktionsbausteins in FUP

Optionale Funktionen:
- Senden von CoLa[i] Kommandos an ein SICK Sensor
- Empfangen von CoLa Antworten eines SICK Sensor
- Empfangen von geräteseitig gesendeten Telegrammen (im SOPAS[ii] Ausgabeformat konfigurierbar)

[i] Die command language (CoLa) ist ein SICK internes 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 als Profinet Controller ein CP-Modul verwenden. Steuerungen mit integrierten Profinet Controller werden nicht unterstützt.

3.2 Unterstützte Feldbus Gateways / Sensoren

Der SICK Sensor kommuniziert über Profinet mit der Steuerung. Sollte der Sensor kein Profinet unterstützen, kann ein CDM425 Gateway-Modul eingesetzt werden.

3.3 Konfiguration in Step7

Bevor der Funktionsbaustein verwendet werden kann, muss in der Step7 Hardwarekonfiguration der entsprechende Sensor bzw. das entsprechende Gateway projektiert werden. Der erste Schritt ist, die entsprechende Gerätestammdatei (GSDML) in die Step7 Hardwarebibliothek zu importieren.

Der Funktionsbaustein ist speziell für den Handshake mode ausgelegt. Bitte nur Module aus der Kategorie „Handshake v2.1“ verwenden, die mit einer Länge zwischen 8...128 Bytes definiert sind. Die verwendeten Adressen dürfen im Peripheriebereich oder außerhalb projektiert werden. Eine Adresszuweisung auf Peripheriebereiche, denen ein Teilprozessabbild mit OB6x-Anbindung (Taktsynchronalarme) zugeordnet ist, darf nicht verwendet werden.

3.3.1 Hardwarekonfiguration

Abbildung *2* zeigt eine Beispielkonfiguration mit einem CDM425 Profinet Gateway.

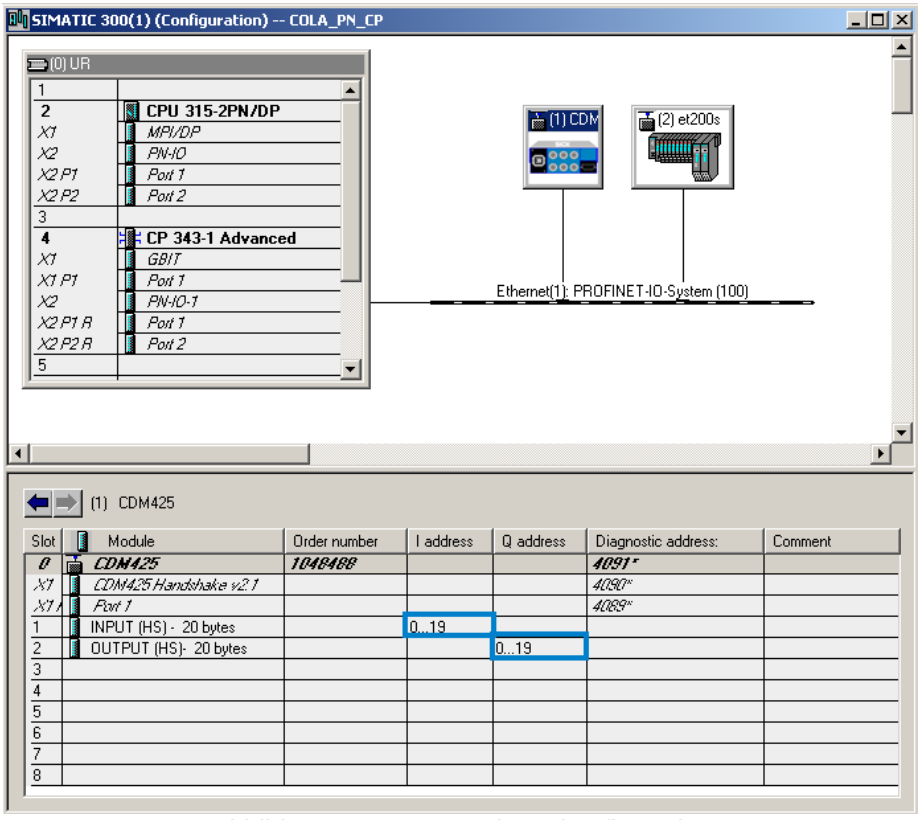

Abbildung 2: Step7 Hardwarekonfiguration

Bitte beachten Sie, dass die E/A-Adressen des CP-Moduls nicht mit den E/A-Adressen der CPU identisch sind, da das CP-Modul einen eigenen Adressbereich besitzt. Aus diesem Grund ist es nicht möglich direkt aus dem S7-Programm mit dem Profinet-Teilnehmer zu kommunizieren.

Der Zugriff auf den kompletten E/A-Bereich des CP-Moduls (hier die Teilnehmer SICK CDM425 und Siemens ET200S) wird über die Funktionen FC11 (PNIO_SEND) und FC12 (PNIO_RECV) realisiert. Diese Funktionen erstellen ein konsistentes E/A-Abbild aller am CP-Modul angeschlossen Geräte.

Um die Siemens FCs zu nutzten, ist es notwendig, dass die projektieren E/A-Bereiche der angeschlossenen Peripherie zusammenhängend konfiguriert sind und bei Adresse 0 anfangen.

3.3.2 Zugriff auf den E/A-Bereich

Der E/A-Bereich der angeschlossenen Teilnehmer sollte jeden SPS-Zyklus ausgelesen bzw. beschrieben werden. In diesem Beispiel werden die Funktionen FC11/FC12 zyklisch im OB1 aufgerufen.

Die Funktion FC12 (PNIO_RECV) muss wie folgt parametriert werden:

CPLADDR:	Hardwareadresse des projektieren CP-Moduls (siehe Abbildung 4)
MODE:	0 = IO-Controller Mode
LEN:	Bytelänge aller Eingangsdaten ab Adresse 0. CDM425 (0..19) = 20 Byte ET200S (20..23) = 4 Byte Insgesamt = 24 Byte
RECV:	Pointer auf einen Datenbereich (DB) wo die eingehenden Daten gespeichet werden sollen.
IOPS:	Pointer auf einen Datenbereich (DB) wo der IOPS (IO Producer Status) gespeichert werden soll. Die Länge des Datenbereichs für die IOPS Daten muss LEN /8 (24/8= 3 Byte) lang sein.

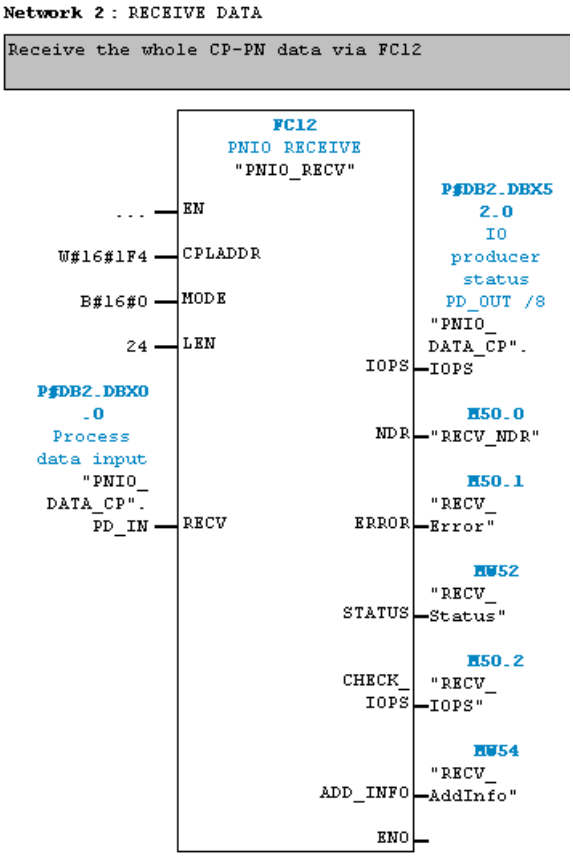

Abbildung 3: Aufruf FC12 (PNIO_RECV) im OB1

Abbildung 4 zeigt, wo man die Hardwareadresse des projektierten CP-Moduls in der Simatic Hardwarekonfiguration finden kann.

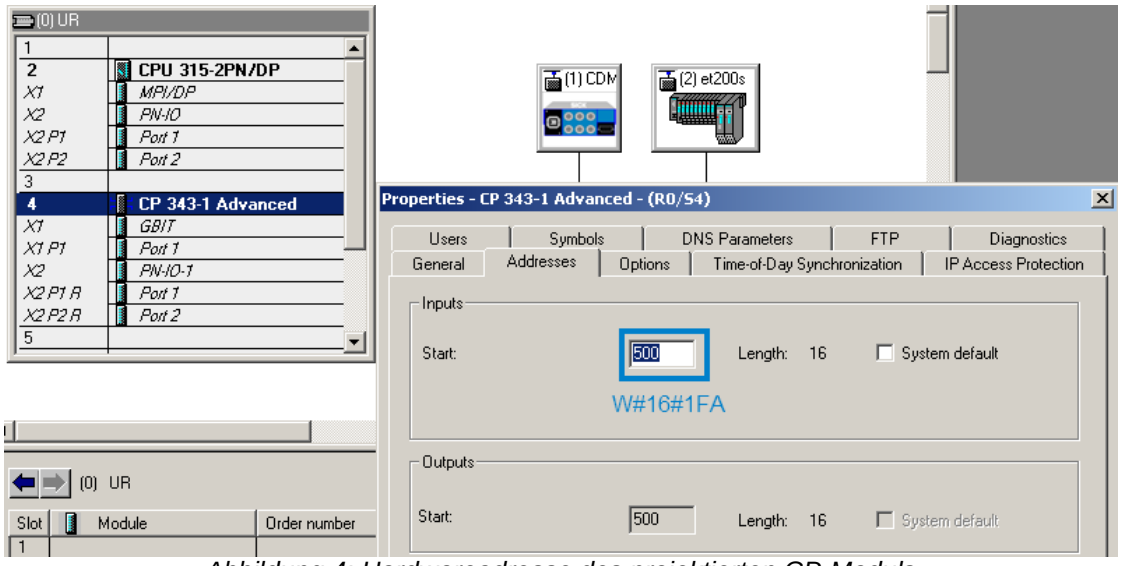

Abbildung 4: Hardwareadresse des projektierten CP-Moduls

Die Funktion FC11 (PNIO_SEND) muss wie folgt parametriert werden:

CPLADDR:	Hardwareadresse des projektieren CP-Moduls (siehe Abbildung 4)
MODE:	0 = IO-Controller Mode
LEN:	Bytelänge aller Ausgangsdaten ab Adresse 0. CDM425 (0..19) = 20 Byte ET200S (20..23) = 4 Byte Insgesamt = 24 Byte
SEND:	Pointer auf einen Datenbereich (DB) wo die zu schreibenden Ausgangsdaten gespeichert sind.
IOCS:	Pointer auf einen Datenbereich (DB) wo der IOCS (IO Consumer Status) gespeichert werden soll. Die Länge des Datenbereichs für die IOCS Daten muss LEN /8 (24/8=3 Byte) lang sein.

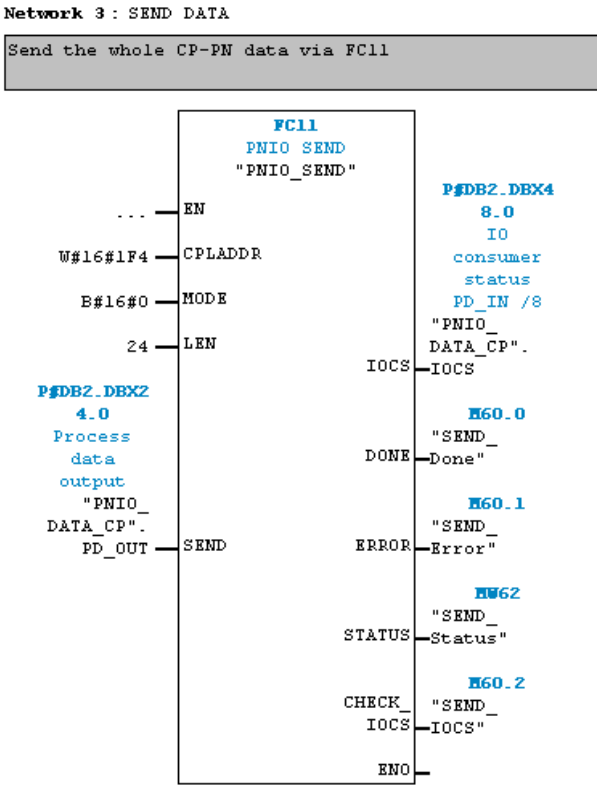

Abbildung 5: Aufruf FC11 (PNIO_SEND) im OB1

4 Bausteinbeschreibung

Der Funktionsbaustein CCOM_PN_CP (FB14) vereinfacht die Benutzung von SICK Sensoren an S7-Steuerungen. Der Baustein ermöglicht das Senden und Empfangen von CoLa Telegrammen über eine in der Hardwarekonfiguration projektierten Profinet Verbindung.

Der Baustein unterstützt bei folgenden Aufgaben:
- Senden von CoLa Kommandos an ein SICK Sensor
- Empfangen von CoLa Antworten eines SICK Sensor
- Empfangen von geräteseitig gesendeten Telegrammen (im SOPAS Ausgabeformat konfigurierbar)

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.

4.1 Bausteinspezifikationen

Bausteinnummer:	FB14
Bausteinname:	CCOM_PN_CP
Version:	1.0
Aufgerufene Bausteine:	SFC20 (BLKMOV) SFB4 (TON)
Verwendete Datenbausteine:	-
Bausteinaufruf:	Zyklisch
Verwendete Merker:	keine
Verwendete Zähler:	keine
Verwendetet Register:	AR1, AR2 (für Multiinstanzen)
Erstellsprache:	Step7-AWL

Die im Funktionsbaustein verwendeten Systembausteine (SFCs / SFBs) müssen auf der jeweils verwendeten Steuerung vorhanden sein.

4.2 Arbeitsweise

Um den CCOM_PN_CP Baustein einsetzten zu können, müssen zunächst die folgenden Parameter angegeben werden.

CPDATA_IN: Pointer auf die Eingangsdaten des Sensors/Gateways. Die Eingangsdaten müssen vorher mit der Funktion FC12 (PNIO_RECV) abgeholt werden.

CPDATA_OUT: Pointer auf die Ausgangsdaten des Sensors/Gateways. Die Ausgangsdaten müssen mit der Funktion FC12 (PNIO_SEND) zum Gerät übertragen werden.

COMMAND: Der Pointer zeigt auf den Datenbereich, in den das CoLa Kommando abgelegt ist. Der Datenbereich muss vom Programmiere selbst erstellt werden (z.B. Datenbaustein mit einem Array of CHAR). Das Kommando muss ohne [STX] / [ETX] Rahmung angegeben werden.

COMMAND_LEN: Zeichenlänge des zu übertragenden CoLa Kommandos.

RECORD: Der Pointer zeigt auf den Datenbereich, in den die vom Gerät gesendeten Telegrame abgelegt werden. Der Datenbereich muss vom Programmierer selbst erstellt werden (z.B. Datenbaustein mit einem Array of BYTE).

4.2.1 Empfangen von Leseergebnissen (RD)

Daten die geräteseitig gesendet werden (RD), werden in den Record geschrieben, sobald der Funktionsbaustein neue Daten empfangen hat. Das Bit RD_DONE zeigt für einen SPS Zyklus den Empfang neuer Daten an. Sobald neue Daten empfangen wurden, wird der Zähler RD_COUNT inkrementiert. Die jeweilige Bytelänge des zuletzt empfangenden Telegramms kann dem Parameter RD_LEN entnommen werden.

4.2.2 Gerätekommunikation über CoLa Kommandos (REQ)

Bei der Kommunikation via CoLa Kommandos, wird das im COMMAND definierte Kommando zum Gerät übertragen. Die resultierende Antwort wird in den vom Pointer RECORD definierten Bereich abgelegt.

Sie starten die Übertragung, indem Sie den Parameter START_REQ mit einer positiven Flanke antriggern. Solange noch keine gültige Antwort auf das gesendete CoLa Kommando eingetroffen ist, wird dies über den Parameter REQ_BUSY signalisiert. Sollte innerhalb der Timeout Zeit (TOUT) keine Antwort eingetroffen sein, wird die Bearbeitung mit einem Timeout Fehler (REQ_ERRORCODE) abgebrochen. Der Ausgangsparameter REQ_DONE zeigt an, dass auf ein CoLa Kommando eine Antwort empfangen wurde (REQ_DONE = TRUE).

4.2.3 Timing

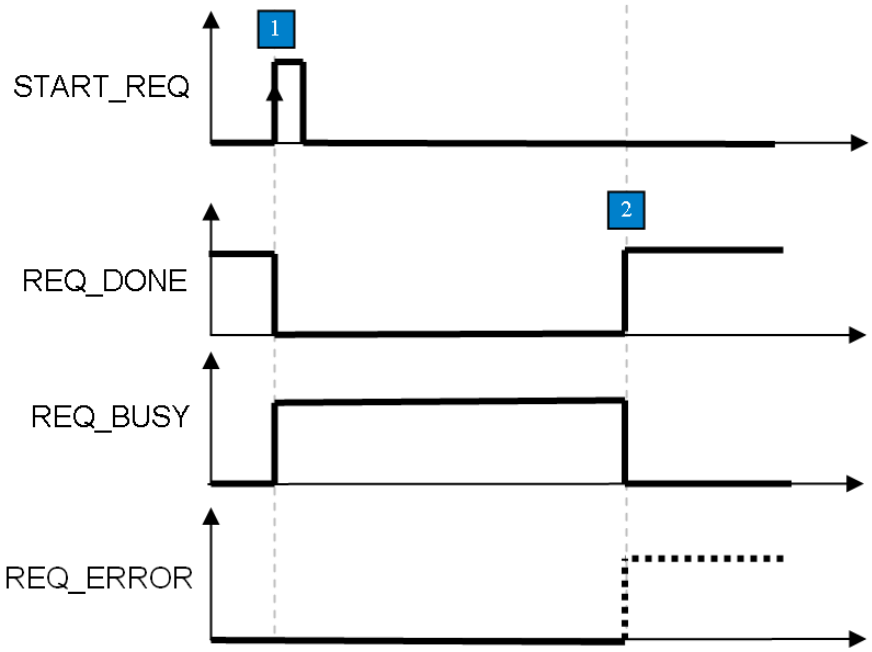

Abbildung 6: Timing Diagramm

1: Anforderung durch eine positve Flanke an START_REQ. Das vom Parameter COMMAND 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 „REQ_DONE“ beendet. Wenn die Aktion fehlerhaft verläuft, wird mit „REQ_ERROR“ beendet. Bei Abbruch mit „REQ_ERROR“ enthält „REQ_ERRORCODE“ den aufgetretenen Fehlercode.

4.3 Verhalten im Fehlerfall

Im Fehlerfall signalisieren die Errorbits REQ_ERROR oder RD_ERROR, dass ein Fehler aufgetreten ist. In diesem Fall wird über den Parameter REQ_ERRORCODE bzw. RD_ERRORCODE ein Fehlercode ausgegeben. Das Errorbit REQ_ERROR bleibt solange gesetzt, bis ein neuer Auftrag gestartet wird. Der Parameter RD_ERROR ist immer nur für einen SPS-Zyklus aktiv und wird danach zurückgesetzt, sollte der Fehler nicht weiter bestehen.

4.4 Rücksetzten der Kommunikation

Über das RESET Bit ist es möglich die Kommunikation zwischen dem Gateway / Sensor und der PLC zurückzusetzen. Hierbei erden die ersten 8 Bytes der Peripherieausgangsseite für eine Sekunde mit null initialisiert. Der Reset-Befehl wird ausgeführt, sobald das RESET = TRUE und START_REQ mit einer positiven Flanke angetriggert wird. Das REQ_BUSY Bit signalisiert, dass der Befehl bearbeitet wird. Ist die Reset-Routine abgeschlossen, wird das REQ_DONE Bit gesetzt.

4.5 Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN	INPUT	BOOL	E,M,D,L, Konst.	Enable
CPDATA_IN	INPUT	ANY	D	Pointer auf den Eingangsbereich des Sensors/Gateways. Es ist nur das Datentyp BYTE zulässig. Hinweis: Beachten Sie, dass der Parameter immer die vollständige Angabe der DB-Parameter erfordert (Bsp.: P#DB13.DBX0.0 BYTE 100). Das Weglassen einer expliziten DB-Nr. ist unzulässig und führt zu einem Bausteinfehler.
CPDATA_ OUT	INPUT	ANY	D	Pointer auf den Ausgangsbereich des Sensors/Gateways. Es ist nur das Datentyp BYTE zulässig. Hinweis: Beachten Sie, dass der Parameter immer die vollständige Angabe der DB-Parameter erfordert (Bsp.: P#DB13.DBX0.0 BYTE 100). Das Weglassen einer expliziten DB-Nr. ist unzulässig und führt zu einem Bausteinfehler.
TOUT	INPUT	TIME	E,M,D,L, Konst.	Zeit, nachdem ein Timeout-Fehler ausgelöst wird. Wenn dieser Parameter nicht beschaltet ist, beträgt die Timeout Zeit standardmäßig 5Sekunden. Bitte beachten Sie, dass einige CoLa Kommandos eine längere Bearbeitungszeit benötigen (z.B. Speicherkommandos).
START_REQ	INPUT	BOOL	E,M,D,L	Positive Flanke: Senden ein CoLa Kommando und wartet auf die entsprechende Antwort.
RESET	INPUT	BOOL	E,M,D,L, Konst.	Rücksetzten der Kommunikation (HS-Counter des Datenflussprotokolls)

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
COMMAND	INPUT	ANY	D	Pointer auf den Bereich, welcher das zu übertragende CoLa Kommando enthält. Es ist nur das Datentyp BYTE zulässig. Das Kommando muss ohne [STX] / [ETX] Rahmung angegeben werden. Hinweis: Beachten Sie, dass der Parameter immer die vollständige Angabe der DB-Parameter erfordert (Bsp.: P#DB13.DBX0.0 BYTE 100). Das Weglassen einer expliziten DB-Nr. ist unzulässig und führt zu einem Bausteinfehler.
COMMAND_ LEN	INPUT	INT	E,M,D,L, Konst.	Anzahl der Bytes, die das zu sendende CoLa Kommando enthält, worauf der Pointer #COMMAND referenziert.
RECORD	INPUT	ANY	D	Pointer auf den Bereich, in denen die vom gerät gesendeten Telegramme abgelegt werden sollen. Es ist nur der Datentyp BYTE zulässig. Hinweis: Beachten Sie, dass der Parameter immer die vollständige Angabe der DB-Parameter erfordert (Bsp.: P#DB13.DBX0.0 BYTE 100). Das Weglassen einer expliziten DB-Nr. ist unzulässig und führt zu einem Bausteinfehler.
RD_DONE	OUTPUT	BOOL	A,M,D,L	Positive Flanke: Ein vom Gerät gesendetes Leseergebnis wurde empfangen (Formatierung siehe SOPAS Ausgabeformat). Wenn ein Leseergebnis empfangen wurde, ist das Bit für jeweils einen SPS-Zyklus gesetzt. Das Leseergebnis steht im vom Parameter #RECORD referenzierten Speicherbereich zur Verfügung.
RD_ COUNT	OUTPUT	BYTE	A,M,D,L	Zählt die Anzahl der Leseergebnisse die empfangen wurden. Der Zähler zählt von 0…255 (Dez.). Bei einem Überlauf fängt der Zähler wieder bei null an.
RD_ LEN	OUTPUT	INT	A,M,D,L	Gibt die Bytelänge des empfangenden Leseergebnisses an.

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
REQ_DONE	OUTPUT	BOOL	A,M,D,L	Zeigt an, ob ein CoLa Kommando abgeschickt und eine Antwort empfangen wurde. TRUE: Bearbeitung abgeschlossen FALSE: Bearbeitung noch nicht abgeschlossen Die Kommandoantwort steht im, vom Parameter #RECORD referenzierten, Speicherbereich zur Verfügung.
REQ_BUSY	OUTPUT	BOOL	A,M,D,L	REQ Auftrag ist in Bearbeitung.
REQ_ LEN	OUTPUT	INT	A,M,D,L	Länge eines Antworttelegramms in BYTE.
REQ_ ERROR	OUTPUT	BOOL	A,M,D,L	REQ Fehler Status: 0: Kein Fehler 1: Abbruch mit Fehler
RD_ERROR	OUTPUT	BOOL	A,M,D,L	RD Fehler Status: 0: Kein Fehler 1: Abbruch mit Fehler
REQ_ERRO RCODE	OUTPUT	WORD	A,M,D,L	REQ Fehlerstatus (siehe Fehlercodes).
RD_ERROR CODE	OUTPUT	WORD	A,M,D,L	RD Fehlerstatus (siehe Fehlercodes).
ENO	OUTPUT	BOOL	A,M,D,L	Enable Ausgang (KOP und FUP).

4.6 Fehlercodes

Die Parameter REQ_ERRORCODE und RD_ERRORCODE enthalten die folgenden Fehlerinformationen:

Fehlercode	Kurzbeschreibung	Beschreibung
W#16#0000	Kein Fehler	Kein Fehler
W#16#0001	Ungültiger Speicherbereich #RECORD Pointer angegeben	Ungültiger Speicherbereich des angegebenen ANY-Pointers. Dem Pointer muss ein DB zugeordnet werden.
W#16#0002	Ungültige Pointerlänge #RECORD angegeben	Die Länge des referenzierten Datenbaustein ist kürzer als die vom Pointer vorgegebene Länge.
W#16#0003	Ungültiger Speicherbereich #COMMAND Pointer angegeben	Ungültiger Speicherbereich des angegebenen ANY-Pointers. Dem Pointer muss ein DB zugeordnet werden.
W#16#0004	Ungültige Pointerlänge #COMMAND angegeben	Die Länge des referenzierten Datenbaustein ist kürzer als die vom Pointer vorgegebene Länge.
W#16#0005	Timeout	Der Auftrag konnte innerhalb der gewählten Timeoutzeit nicht ausgeführt werden. Dies könnte folgende Ursache haben: - Gerät ist nicht mit der SPS Verbunden - Kommunikationsparameter fehlerhaft - Verwendung von CoLa Kommandos, die keine Antwort (Echo) zurücksenden - Bearbeitungszeit des Kommandos > Timeout Zeit
W#16#0006	Ungültige Kommandolänge	Die Länge des zu sendenden Kommandos ist länger als die angegebene Kommandolänge (COMMAND_LEN).
W#16#0007	SFC20 Fehler	Der Baustein SFC20 (BLKMOV) meldet einen Bausteinfehler. Der Fehlercode wir in der Variable „nStatusBLKMOV“ des instanziierten Datenbausteins angezeigt. Zur Interpretation des Fehlercodes verwenden Sie bitte das Step7 Hilfesystem.
W#16#000A	Empfangendes Telegramm > #RECORD Länge	Das empfangende Telegram ist länger als die angegebene #RECORD Länge.
W#16#000B	Ungültiges Input Modul	Die projektierte Länge des Input Moduls ist ungültig (Pointerlänge CPDATA_IN). Gültiger Wertebereich: [8..128]
W#16#000C	Ungültiges Output Modul	Die projektierte Länge des Output Moduls ist ungültig (Pointerlänge CPDATA_OUT). Gültiger Wertebereich: [8..128]
W#16#000D	Interner Bausteinfehler	Interner Bausteinfehler

5 Beispiel

Abbildung 7 zeigt eine Beispielbeschaltung des CCOM_PN_CP FBs. In der Hardwarekonfiguration ist ein SICK Gerät mit einer Prozessdatenbreite von 20 Byte Input/Output projektiert. Zusätzlich zum SICK Gerät ist am CP-Modul noch ein weiteres Gerät projektiert. Der E/A-Bereich der beiden Geräte wird mit den Funktionen FC11/FC12 beschrieben bzw. ausgelesen (siehe Kapitel 3).

Programmaufruf:

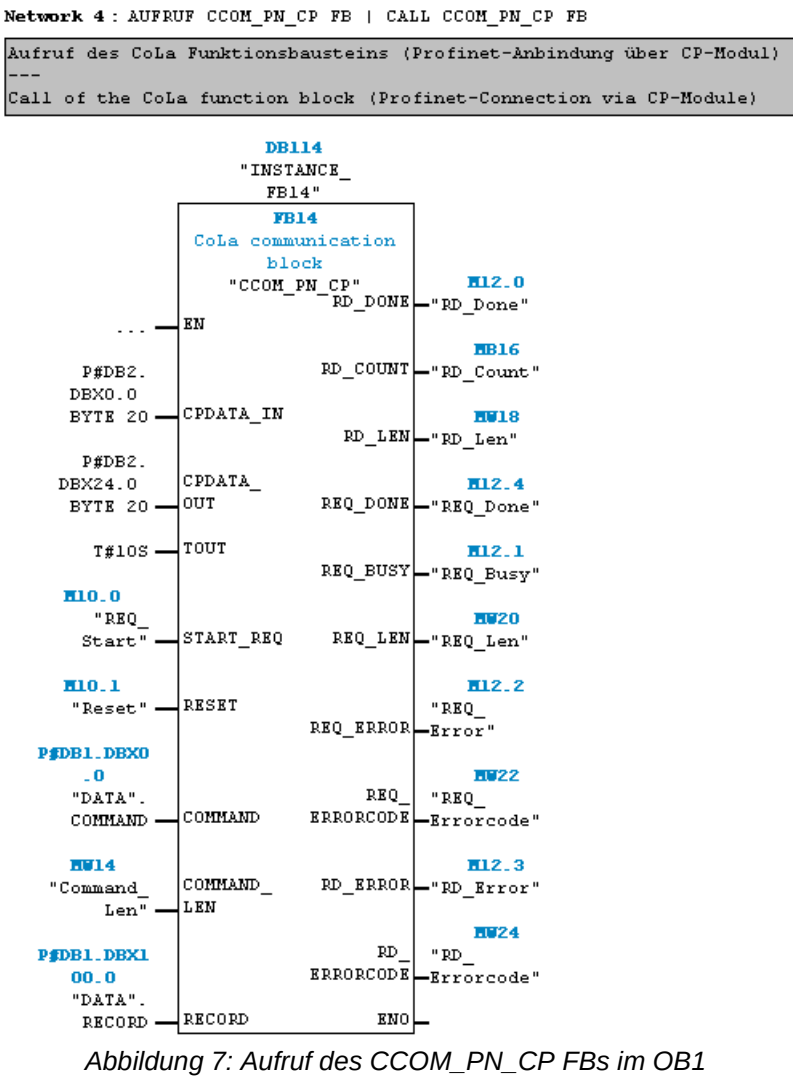

Abbildung 7: Aufruf des CCOM_PN_CP FBs im OB1

Variablentabelle zum ausführen eines CoLa Kommandos:

// CCOM PN CP function block				
M 10.0	"REQ_Start"	BOOL	true	
M 10.1	"Reset"	BOOL	false	
// Reading Result Status				
M 12.0	"RD_Done"	BOOL	false	
MB 16	"RD_Count"	DEZ	101	
M 12.3	"RD_Error"	BOOL	false	
MW 24	"RD_Errorcode"	HEX	W#16#0000	
// Requesting Result Status				
M 12.4	"REQ_Done"	BOOL	true	
M 12.1	"REQ_Busy"	BOOL	false	
M 12.2	"REQ_Error"	BOOL	false	
MW 22	"REQ_Errorcode"	HEX	W#16#0000	
// Command				
MW 14	"Command_Len"	DEZ	13	13
DB1.DBB 0	"DATA".COMMAND[1]	ZEICHEN	's'	's'
DB1.DBB 1	"DATA".COMMAND[2]	ZEICHEN	'M'	'M'
DB1.DBB 2	"DATA".COMMAND[3]	ZEICHEN	'N'	'N'
DB1.DBB 3	"DATA".COMMAND[4]	ZEICHEN	' '	' '
DB1.DBB 4	"DATA".COMMAND[5]	ZEICHEN	'm'	'm'
DB1.DBB 5	"DATA".COMMAND[6]	ZEICHEN	'T'	'T'
DB1.DBB 6	"DATA".COMMAND[7]	ZEICHEN	'C'	'C'
DB1.DBB 7	"DATA".COMMAND[8]	ZEICHEN	'g'	'g'
DB1.DBB 8	"DATA".COMMAND[9]	ZEICHEN	'a'	'a'
DB1.DBB 9	"DATA".COMMAND[10]	ZEICHEN	't'	't'
DB1.DBB 10	"DATA".COMMAND[11]	ZEICHEN	'e'	'e'
DB1.DBB 11	"DATA".COMMAND[12]	ZEICHEN	'o'	'o'
DB1.DBB 12	"DATA".COMMAND[13]	ZEICHEN	'n'	'n'
DB1.DBB 13	"DATA".COMMAND[14]	ZEICHEN	B#16#00	
DB1.DBB 14	"DATA".COMMAND[15]	HEX	B#16#00	
DB1.DBB 15	"DATA".COMMAND[16]	ZEICHEN	B#16#00	
DB1.DBB 16	"DATA".COMMAND[17]	ZEICHEN	B#16#00	
DB1.DBB 17	"DATA".COMMAND[18]	ZEICHEN	B#16#00	
DB1.DBB 18	"DATA".COMMAND[19]	ZEICHEN	B#16#00	
DB1.DBB 19	"DATA".COMMAND[20]	ZEICHEN	B#16#00	

Das CoLa Kommando (hier: 'sMN mTCgateon') wird ausgeführt, sobald das Bit „REQ_START“ mit einer positiven Flanke angesteuert wird. Dem Parameter Command_Len wird die Länge des Kommandos übergeben (hier: 13 Zeichen).

Variablentabelle für eingehende Kommandoantworten:

// Record

MW 18	"RD_Len"	DEZ	31
MW 20	"REQ_Len"	DEZ	15
DB1.DBB 100	"DATA".RECORD[1]	ZEICHEN	's'
DB1.DBB 101	"DATA".RECORD[2]	ZEICHEN	'A'
DB1.DBB 102	"DATA".RECORD[3]	ZEICHEN	'N'
DB1.DBB 103	"DATA".RECORD[4]	ZEICHEN	' '
DB1.DBB 104	"DATA".RECORD[5]	ZEICHEN	'm'
DB1.DBB 105	"DATA".RECORD[6]	ZEICHEN	'T'
DB1.DBB 106	"DATA".RECORD[7]	ZEICHEN	'C'
DB1.DBB 107	"DATA".RECORD[8]	ZEICHEN	'g'
DB1.DBB 108	"DATA".RECORD[9]	ZEICHEN	'a'
DB1.DBB 109	"DATA".RECORD[10]	ZEICHEN	't'
DB1.DBB 110	"DATA".RECORD[11]	ZEICHEN	'e'
DB1.DBB 111	"DATA".RECORD[12]	ZEICHEN	'o'
DB1.DBB 112	"DATA".RECORD[13]	ZEICHEN	'n'
DB1.DBB 113	"DATA".RECORD[14]	ZEICHEN	' '
DB1.DBB 114	"DATA".RECORD[15]	ZEICHEN	'1'
DB1.DBB 115	"DATA".RECORD[16]	ZEICHEN	B#16#00
DB1.DBB 116	"DATA".RECORD[17]	ZEICHEN	B#16#00
DB1.DBB 117	"DATA".RECORD[18]	ZEICHEN	B#16#00
DB1.DBB 118	"DATA".RECORD[19]	ZEICHEN	B#16#00
DB1.DBB 119	"DATA".RECORD[20]	ZEICHEN	B#16#00

Die Kommandoantwort (REQ) auf ein gesendete Kommando (hier: ’sAN mTCgateon 1’) steht im Record Bereich zur Verfügung, sobald sich der Wert des Ausgangsbits „REQ_DONE“ von FALSE auf TRUE ändert (positive Flanke). Der Parameter „REQ_LEN“ gibt an, wie viele Bytes empfangen wurden bzw. gültig sind.

Variablentabelle für eingehende Leseergebnisse:

// Record

MW 18	"RD_Len"	DEZ	9
MW 20	"REQ_Len"	DEZ	15
DB1.DBB 100	"DATA".RECORD[1]	ZEICHEN	'1'
DB1.DBB 101	"DATA".RECORD[2]	ZEICHEN	'2'
DB1.DBB 102	"DATA".RECORD[3]	ZEICHEN	'3'
DB1.DBB 103	"DATA".RECORD[4]	ZEICHEN	'4'
DB1.DBB 104	"DATA".RECORD[5]	ZEICHEN	'5'
DB1.DBB 105	"DATA".RECORD[6]	ZEICHEN	'6'
DB1.DBB 106	"DATA".RECORD[7]	ZEICHEN	'7'
DB1.DBB 107	"DATA".RECORD[8]	ZEICHEN	'8'
DB1.DBB 108	"DATA".RECORD[9]	ZEICHEN	'9'
DB1.DBB 109	"DATA".RECORD[10]	ZEICHEN	't'
DB1.DBB 110	"DATA".RECORD[11]	ZEICHEN	'e'
DB1.DBB 111	"DATA".RECORD[12]	ZEICHEN	'o'
DB1.DBB 112	"DATA".RECORD[13]	ZEICHEN	'n'
DB1.DBB 113	"DATA".RECORD[14]	ZEICHEN	' '
DB1.DBB 114	"DATA".RECORD[15]	ZEICHEN	'1'
DB1.DBB 115	"DATA".RECORD[16]	ZEICHEN	B#16#00
DB1.DBB 116	"DATA".RECORD[17]	ZEICHEN	B#16#00
DB1.DBB 117	"DATA".RECORD[18]	ZEICHEN	B#16#00
DB1.DBB 118	"DATA".RECORD[19]	ZEICHEN	B#16#00
DB1.DBB 119	"DATA".RECORD[20]	ZEICHEN	B#16#00

Daten die geräteseitig gesendet werden (RD), werden in den Record geschrieben, sobald der Funktionsbaustein neue Daten empfangen hat. Das Bit „RD_DONE“ zeigt für einen SPS Zyklus den Empfang neuer Daten an (Signalwechsel von FALSE auf TRUE). Sobald neue Daten empfangen wurden, wird der Zähler RD_COUNT inkrementiert. Der Parameter „RD_LEN“ gibt an, wie viele Bytes empfangen wurden bzw. gültig sind.