Technische Information

SICK
RFH6xx Funktionsbaustein

Bausteinversion V2.X

SICK RFH6XX PNDP Funktionsbaustein für
Siemens S7 Steuerungen (Step7 V5.5)

SICK
Sensor Intelligence.

Inhaltsverzeichnis

1 Zu diesem Dokument

Bitte lesen Sie dieses Kapitel sorgfältig, bevor Sie mit dieser Betriebsanleitung und den SICK RFH6XX Funktionsbaustein arbeiten.

1.1 Funktion dieses Dokuments

Diese Betriebsanleitung beschreibt den Umgang mit dem SICK RFH6XX 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 RFH6XX PNDP“ wird zur Kommunikation zwischen einer SIMATIC Steuerung und einem SICK RFH6XX RFID Gerät verwendet.

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

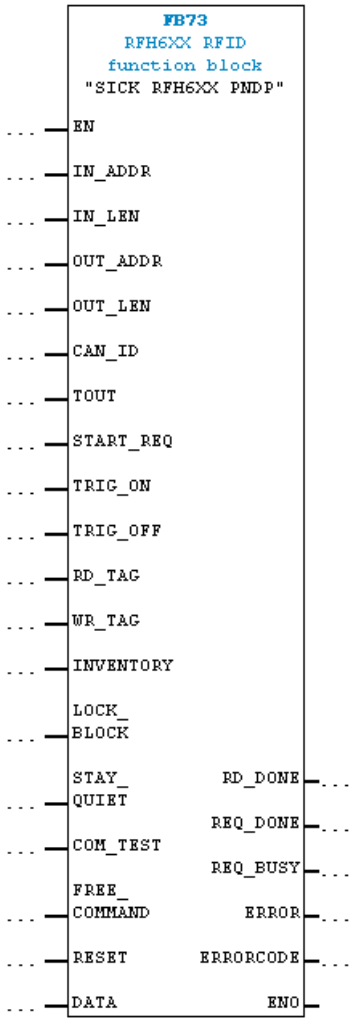

Abbildung 1: SICK RFH6XX PNDP Funktionsbaustein

Bausteinfunktionalitäten:

- Senden eines Trigger Befehls (CoLa[i] Kommando) über die SPS
- Empfang von Leseergebnissen (im SOPAS-ET[ii] Ausgabeformat definiert)
- Lesen und Schreiben von Transponder-Inhalten
- Ausführen eines Inventory-Befehls (Anzeige aller Transpondern im Lesefeld)
- Permanentes sperren von Transponder Blöcken
- Ausführen eines Kommunikationstests
- 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 und der 400er Familie betrieben werden. Es werden nur Steuerungen unterstützt, die die verwendete Feldbusschnittstelle integriert haben. Die Kommunikation über einen Kommunikationsprozessor wird von diesem Baustein 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 unterstützen, können Gateway-Module eingesetzt werden.

Folgenden Gateways werden vom Funktionsbaustein unterstützt:
- CDM 425 (Profinet), ab Firmware Version V3.31
- CDF 600 (Profibus), ab Firmware Version V1.15
- CDM 420 inkl. CMF400 Profibus Modul, ab Firmware Version V1.100

Notwendige RFH Firmware Version:
- RFH6xx, ab Firmware Version V1.31

3.3 Konfiguration in Step7

Bevor der Funktionsbaustein verwendet werden kann, muss in der Step7 Hardwarekonfiguration der RFH entsprechend projektiert werden. Hierfür muss die entsprechende Gerätestammdatei (GSD-Datei) in die Step7 Hardwarebibliothek importiert werden.

Der Funktionsbaustein ist speziell für den Handshake-Mode (Confirmed Messaging Protocol) ausgelegt. Bitte nur HS-Module mit einer Länge zwischen 8…128 Bytes verwenden. 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.

Abbildung 2 zeigt eine Beispielprojektierung des RFHs in Verbindung mit einem CDF600 Profibus Gateway.

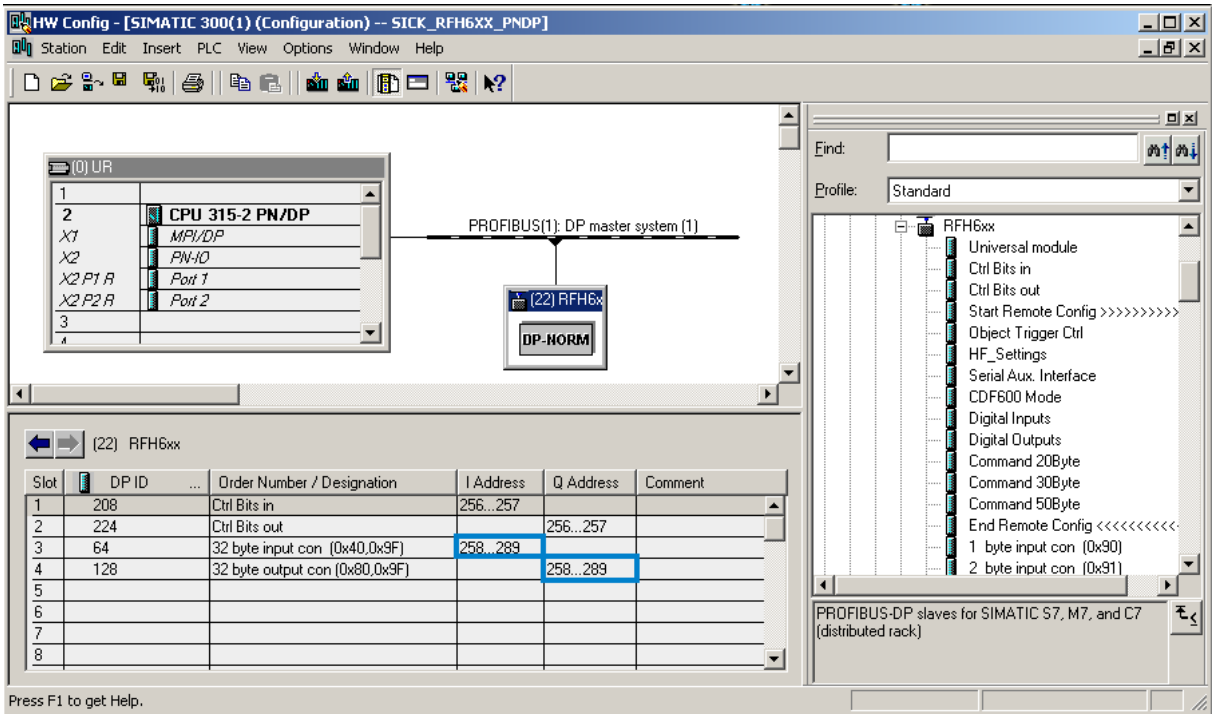

Abbildung 2: Step7 Hardwarekonfigurationsbeispiel

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 RFH Baustein kapselt den Funktionsbaustein „SICK CCOM PNDP“ (FB10), der die Kommunikation zwischen SPS und Sensor ermöglich. Der FC10 (SICK COLA ACCESS) wird intern zur Interpretation der CoLa-Telegramme verwendet.

4.1 Bausteinspezifikationen

Bausteinnummer:	FB73
Bausteinname:	SICK RFH6XX PNDP
Version:	2.1
Aufgerufene Bausteine:	SFC 14 (DPRD_DAT) SFC 15 (DPWR_DAT) SFC20 (BLKMOV) SFB4 (TON) FB10 (SICK CCOM PNDP) FC10 (SICK COLA ACCESS)
Verwendete Datenbausteine:	DB73 (SICK RFH DATA)
Bausteinaufruf:	Zyklisch
Verwendete Merker:	keine
Verwendete Zähler:	keine
Verwendetet Register:	AR1, AR2 (für Multiinstanzaufruf)
Muliinstanzfähig:	ja
Erstellsprache:	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 RFH6XX PNDP Baustein aktualisiert werden.

4.2 Arbeitsweise

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

IN_ADDR: Projektierte Anfangsadresse aus dem E-Bereich des verwendeten Input-Moduls. Die Eingangsadresse wird mit der Hardwareprojektierung festgelegt (siehe Kapitel 3.3). Der Wert muss im Hexadezimalformat angegeben werden (z.B. Adresse 256 = W#16#100).

IN_LEN: Länge des verwendeten Input-Moduls in der Hardwarekonfiguration. Die Länge des Eingabemoduls wird mit der Hardwareprojektierung festgelegt (siehe Kapitel 3.3).

OUT_ADDR: Projektierte Anfangsadresse aus dem A-Bereich des verwendeten Output-Moduls. Die Ausgangsadresse wird mit der Hardwareprojektierung festgelegt (siehe Kapitel siehe Kapitel 3.3). Der Wert muss im Hexadezimalformat angegeben werden (z.B. Adresse 256 = W#16#100).

OUT_LEN: Länge des verwendeten Output-Moduls in der Hardwarekonfiguration. Die Länge des Ausgabemoduls wird mit der Hardwareprojektierung festgelegt (siehe Kapitel siehe Kapitel 3.3).

DATA: Der zum Funktionsbaustein gehörende Datenbaustein (DB73) beinhaltet Ein- und Ausgabeparameter der unterstützten Bausteinaktionen. Der Datenbaustein muss dem Eingangsparameter „DATA“ des Funktionsbausteins übergeben werden.

Ausführbare Bausteinfunktionen:

- Trigger on → Öffnet das Lesetor des Gerätes per CoLa Kommando
- Trigger off → Schließt das Lesetor des Gerätes per CoLa Kommando
- Read Tag → Auslesen von Transponderdaten
- Write Tag → Schreiben von Transponderdaten
- Inventory → Die Inventory Aktion sucht im Lesebereich des RFHs nach aktiven Transpondern und gibt deren UIDs zurück.
- Lock Block → Permanentes sperren eines gewählten Transponderblocks
- Stay Quiet → Stummschalten eines im Feld befindenden RFID-Tags.
- Kommunikationstest → Prüft, ob das Gerät per „sRI0“ Kommando erreichbar ist
- Free Command → Ausführen eines frei wählbaren CoLa Kommandos
- Reset → Reset der Kommunikation

Um eine Bausteinaktion (TRIG_ON, RD_TAG, 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. RD_TAG) 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 (ReadingRe-

sult.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.

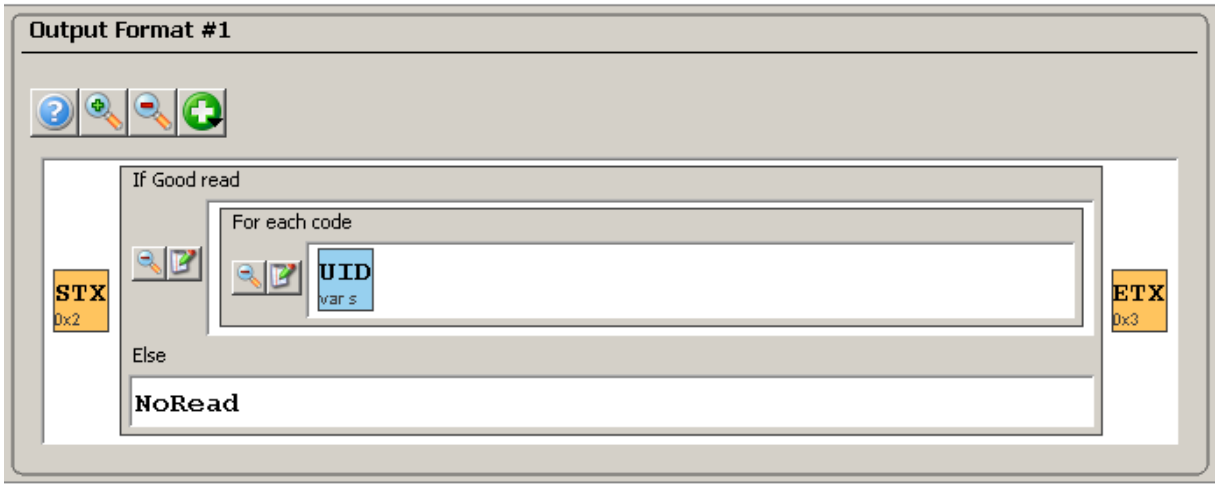

Abbildung 3: SOPAS Ausgabeformat

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.

Über das RESET Bit ist es möglich, die Kommunikation zwischen dem Sensor und der SPS zurückzusetzen. Der Reset-Befehl wird ausgeführt, sobald das RESET Bit vorgewählt und das START_REQ Bit mit einer positiven Flanke (Signalwechsel von null auf eins) angetriggert wird. Das REQ_BUSY Bit signalisiert, dass der Befehl bearbeitet wird. Ist die Reset-Routine abgeschlossen, wird das REQ_DONE Bit gesetzt.

Durch das Rücksetzen werden folgende Aktionen ausgeführt:
- Rücksetzen der Counter vom Confirmed Messaging Protokoll (Gerätekommunikation)
- Rücksetzen aller Fehlermeldungen

4.4 Timing

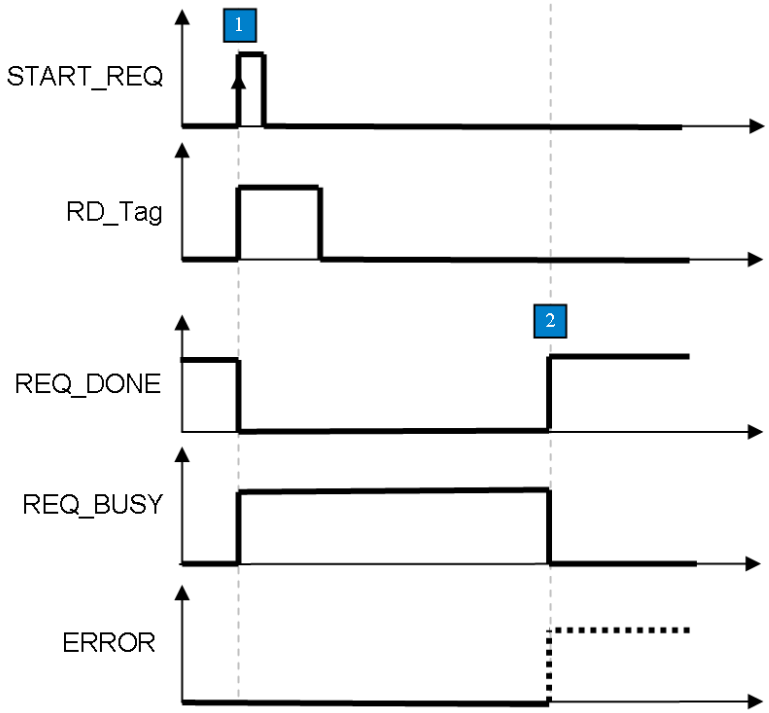

Abbildung 4: Timing Diagramm

1: Anforderung durch Pos Flanke an START_REQ
Die gewünschte Aktion (hier RD_Tag) 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 RFH DATA“ (DB73) 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).

DB73 -- "SICK RFH DATA" -- SICK_RFH6XX_PNDP\SIMATIC 300(1)\CPU 315-2 PN/DP\...\DB73

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	Mode	STRUCT		-- MODE --
+0.0	bMode	BOOL	FALSE	1: Use a fixed UID | 0: Use the UID of the transponder in the field (IN)
+2.0	arrUID	ARRAY[1..8]		If bMode=1, this UID will be used for a Read/Write/Lock/Stay quit job (IN/OUT)
*1.0		BYTE		
=10.0		END_STRUCT		
+10.0	iLockBlock	INT	0	Number of the block that should be locked (IN)
+12.0	Inventory	STRUCT		-- INVENTORY --
+0.0	iNumRetTags	INT	0	Number of returned transponders (OUT)
+2.0	arrTagInfo	ARRAY[1..5]		Max. 5 transponder (OUT)
*0.0		STRUCT		
+0.0	nError	BYTE	B#16#0	Error code (OUT)
+1.0	nRSSI	BYTE	B#16#0	RSSI RX value (OUT)
+2.0	nDSFID	BYTE	B#16#0	DSFID (OUT)
+4.0	arrUID	ARRAY[1..8]		UID (OUT)
*1.0		BYTE		
=12.0		END_STRUCT		
=62.0		END_STRUCT		
+74.0	ReadTag	STRUCT		-- READ TAG --
+0.0	iStartBlock	INT	0	Number of the first block that should be read (IN)
+2.0	iNumBlocks	INT	0	Number of blocks that should be read (IN)
+4.0	iDataLength	INT	0	Content length in bytes (OUT)
+6.0	arrData	ARRAY[1..128]		Data to be read (OUT)
*1.0		BYTE		
=134.0		END_STRUCT		
+208.0	WriteTag	STRUCT		-- WRITE TAG --
+0.0	iStartBlock	INT	0	Number of the first block that should be written (IN)
+2.0	iNumBlocks	INT	0	Number of blocks that should be written (IN)
+4.0	iBlockSize	INT	4	Block size in bytes (IN)
+6.0	arrData	ARRAY[1..128]		Data to be write (IN)
*1.0		BYTE		
=134.0		END_STRUCT		
+342.0	FreeCommand	STRUCT		-- FREE COMMAND --
+0.0	iCommandLength	INT	0	Byte 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		
+546.0	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		
=750.0		END_STRUCT		

Abbildung 5: Struktur des SICK RFH DATA Nutzdaten DBs

4.5.1 Mode

Der RFH kann immer nur mit einem Transponder gleichzeitig kommunizieren. Aus diesem Grund werden Lese- und Schreibbefehle immer adressiert ausgeführt. Zum Identifizieren des Transponders verwendet der FB die UID (Unique Identifier).

Um zu bestimmen, mit welcher Transponder UID kommuniziert werden soll, unterstützt der Funktionsbaustein zwei Modi:

Mode 1: Es wird immer mit dem Transponder kommuniziert, der sich aktuell im Lesefeld befindet. Dieser Modus kann nur eingesetzt werden, wenn sich genau ein Tag im Feld befindet.

Mode 2: Es wird eine, vom Anwender definierte Transponder-UID zur Kommunikation verwendet.

Parameter	Deklaration	Datentyp	Beschreibung
Mode.bMode	Input	BOOL	Adressierungsmodus FALSE: Mode 1 aktiv TRUE: Mode 2 aktiv
Mode.arrUID	Input/Output	INT	Transponder Identifikation (UID) *Im Mode 1 wird die UID automatisch ausgelesen*

Tabelle 1: Mode Parameter

4.5.2 Lock block

Mit Hilfe der Lock Block Aktion hat man die Möglichkeit einen beliebigen Block auf dem RFID Tag gegen Wiederbeschreiben zu schützen. Die Blocknummer gibt man über den Parameter iLockBlock an, bevor man die Funktionsbaustein-Aktion ausführt. Die Aktion sperrt den gewählten block permanent. Ein entsperren ist nicht möglich.

Parameter	Deklaration	Datentyp	Beschreibung
iLockBlock	INPUT	INT	Nummer des Blocks, der gegen Wiederbeschreiben geschützt werden soll.

4.5.3 Inventory

Die Inventory Aktion sucht im Empfangsbereich des Sensors nach aktiven Transpondern. Für jeden erkannten Transponder (max. 5 Transponder) stellt der FB die folgenden Informationen zur Verfügung.

Parameter	Deklaration	Datentyp	Beschreibung
Inventory. iNumRetTags	Output	INT	Anzahl der erkannten Transponder
Inventory. arrTagInfo[ ].nError	Output	BYTE	Transponder Errorcode (siehe RFH Betriebsanleitung)
Inventory. arrTagInfo[ ].nRSSI	Output	BYTE	RSSI (Signalstärke des erkannten Transponders)
Inventory. arrTagInfo[ ].nDSFID	Output	BYTE	DSFID der erkannten Transponder
Inventory. arrTagInfo[ ].arrUID	Output	ARRAY [1..8] OF BYTE	UID der erkannten Transponder im HEX-Format

4.5.4 Read Tag

Die Read Tag Aktion liest einen definierten Datenbereich eines Tags aus. Die Aktion ist immer nur auf einen Tag anwendbar. Mit welchem Transponder kommuniziert werden soll, ist vom gewählten Modus abhängig (siehe Kapitel 4.5.1).

Vor jedem Lesevorgang muss angegeben werden welche Blöcke auf dem Transponder ausgelesen werden sollen. Nach einem erfolgreichen Lesevorgang wird die Bytelänge der gelesenen Daten, sowie die Nutzdaten im Nutzdaten DB abgelegt.

Parameter	Deklaration	Datentyp	Beschreibung
ReadTag. iStartBlock	Input	INT	Blocknummer, bei dem der Lesevorgang gestartet werden soll
ReadTag. iNumBlocks	Input	INT	Anzahl der Blöcke die ausgelesen werden sollen
ReadTag. iDataLength	Output	INT	Länge des gelesenen Inhalts in Byte
ReadTag.arrData	Output	ARRAY [1..128] OF BYTE	Inhalt der gelesenen Blöcke

Tabelle 2: Read Tag Parameter

4.5.5 Write Tag

Die Write Tag Funktion schreibt auf einen definierten Datenbereich eines Tags. Die Aktion ist immer nur auf einen Tag anwendbar. Mit welchem Transponder kommuniziert werden soll, ist vom gewählten Modus abhängig (siehe Kapitel 4.5.1).

Vor jeden Schreibevorgang muss angegeben werden, ab welchem Block der Schreibbefehl startet und wie viele Blöcke geschrieben werden sollen. Da die Blocklänge eines Transponders sich je nach Tag-Typ ändern kann, muss diese ebenfalls mit angegeben werden (siehe Informationen des Tag-Herstellers).

Parameter	Deklaration	Datentyp	Beschreibung
WriteTag. iStartBlock	Input	INT	Blocknummer, bei den der Schreibevorgang gestartet werden soll
WriteTag. iNumBlocks	Input	INT	Anzahl der Blöcke die geschrieben werden sollen.
WriteTag. iBlockSize	Input	INT	Bytegröße eines Blocks Gültiger Wertebereich: [4,8,12,16,…]
WriteTag.arrData	Input	ARRAY [1..128] OF BYTE	Daten die in die Transponderblöcke geschrieben werden sollen.

Tabelle 3: Write Tag Parameter

4.5.6 Free Command

Mit Hilfe des freien Kommandos hat man die Möglichkeit über ein gültiges CoLa Kommando mit dem RFH 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 4: Free Command Parameter

4.5.7 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 dass Vorgehen beim Empfang von längeren Datentelegrammen.

Tabelle 5: 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 RFH DATA Datenbaustein:
Im mitgelieferten Nutzdatenbaustein (DB73) 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 6: Empfangen von Leseergebnissen > 200 Bytes (Änderung im Datenbaustein)

Änderung im SICK RFH6XX PNDP 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 500 Bytes nicht unterschreiten, muss aber größer/gleich der ReadingResult.arrResult Länge sein.

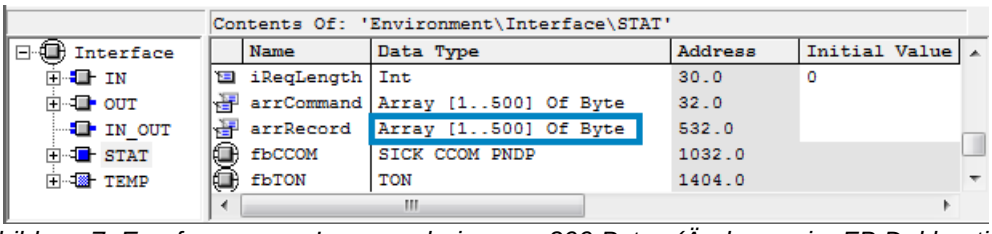

Abbildung 7: Empfangen von Leseergebnissen > 200 Bytes (Änderung im FB Deklaration)

Im Netzwerk 3 des SICK RFH6XX PNDP 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     500
      T     #iArrayRecLen

//-- LENGTH OF THE COMMAND ARRAY
      L     500
      T     #iArrayComLen

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

//-- FRAMING
      SET                                   // Add framing
      =     #bAddFraming

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

Abbildung 8: 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)
IN_ADDR	INPUT	WORD	E,M,D,L, Konst.	Projektierte Anfangsadresse aus dem E-Bereich des gewählten Moduls.
IN_LEN	INPUT	INT	E,M,D,L, Konst.	Länge des verwendeten Input-Moduls in der Hardwarekonfiguration. Gültiger Wertebereich: [8..128]
OUT_ADDR	INPUT	WORD	E,M,D,L, Konst.	Projektierte Anfangsadresse aus dem A-Bereich des gewählten Moduls.
OUT_LEN	INPUT	INT	E,M,D,L, Konst.	Länge des verwendeten Output-Moduls in der Hardwarekonfiguration. Gültiger Wertebereich: [8..128]
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 (DB73) abgelegt.

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
RD_TAG	INPUT	BOOL	E,M,D,L, Konst.	Bausteinaktion: Auslesen von Tag Inhalten. Die Aktion setzt voraus, dass die Parameter der Struktur „ReadTag“ des übergebenen Datenbausteins mit gültigen Werten belegt sind (siehe Kapitel 4.5.4). Welcher Transponder ausgelesen werden soll ist vom gewählten Adressierungsmodus abhängig (siehe Kapitel 4.5.1).
WR_TAG	INPUT	BOOL	E,M,D,L, Konst.	Bausteinaktion: Schreiben von Tag Inhalten. Die Aktion setzt voraus, dass die Parameter der Struktur „WriteTag“ des übergebenen Datenbausteins die Parameter mit gültigen Werten belegt sind (siehe Kapitel 4.5.5). Welcher Transponder beschrieben werden soll ist vom gewählten Adressierungsmodus abhängig (siehe Kapitel 4.5.1).
INVENTORY	INPUT	BOOL	E,M,D,L, Konst.	Sucht im Empfangsbereich nach aktiven Transpondern und gibt deren UID, DSFID und die RSSI Signalstärke zurück.
LOCK_ BLOCK	INPUT	BOOL	E,M,D,L, Konst.	Schützt einen definierten Block gegen Wiederbeschreiben. Die Aktion setzt voraus, dass der Parameter iLockBlock im übergebenen Datenbaustein mit einer gültigen Blocknummer belegt ist (siehe Kapitel 4.5.5). Die Aktion sperrt den gewählten Block permanent. Ein entsperren ist nicht möglich.
STAY_QUIT	INPUT	BOOL	E,M,D,L, Konst.	Stummschalten des im Feld befindenden RFID-Tags. Die Aktion kann nur verwendet werden, wenn das HF-Feld des RFID Geräts dauerhaft eingeschaltet ist (siehe SOPAS → Transponderkommunikation → HF-Feld).

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
COM_TEST	INPUT	BOOL	E,M,D,L, Konst.	Bausteinaktion: Ausführen eines Kommunikationstests. REQ_DONE= TRUE: Kommunikation OK REQ_DONE= FALSE: Kommunikation nicht OK
FREE_ COMMAND	INPUT	BOOL	E,M,D,L, Konst.	Bausteinaktion: Ausführen eines freien Kommandos. Die Aktion setzt voraus, dass im Nutzdatenbaustein (DB73) in der Struktur (FreeCommand) die Parameter iCommandLength und arrCommand mit gültigen Daten belegt sind (siehe Kapitel 4.5.6). Die Kommandoantwort steht nach einer erfolgreichen Übertragung (REQ_DONE=TRUE) im RESULT Bereich des Datenbausteins zur Verfügung.
RESET	INPUT	BOOL	E,M,D,L, Konst.	Bausteinaktion: Rücksetzten der Kommunikation zum Gerät.
DATA	INPUT	BLOCK_ DB	Konst.	Übergabe des zugehörigen Nutzdatenbausteins, der für die Parametrierung der Bausteinfunktionen sowie für das Ablegen des Leseergebnisses benötigt wird (DB73).
RD_DONE	OUTPUT	BOOL	A,M,D,L	Positive Flanke: Neues Leseergebnis empfangen
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 6: 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: - Ungültige E/A Adressen - Ungültige Länge der E/A Adressen - Es wurde ein Telegramm > arrRecord empfangen
W#16#**XX**0A	Gerätefehler	Es ist ein Gerätefehler aufgetreten (’sFA XX’) **XX** = Gerätefehler (siehe Gerätedokumentation)

Fehlercode	Kurzbeschreibung	Beschreibung
W#16#000B	Ungültige Kommando-antwort	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“ - Tag nicht lang genug im Feld - Zugriff auf einen nicht existierenden Tag-Bereich (iStartBlock und iNumBlocks Parameter prüfen) - Ungültige UID (Mode.arrUID prüfen)
W#16#000C – W#16#000F	Reserviert	Reserviert
W#16#0010	Tags im Feld > 5 (Inventory)	Inventory kann nicht ausgeführt werden, da sich mehr als 5 Transponder im Lesefeld des RFHs befinden.
W#16#0011	ReadTag.iStartBlock < 0	Ungültiger Lesebeginn (Read Tag)
W#16#0012	32 < ReadTag. iNumBlocks <= 0	Pro Aktionsaufruf können maximal 128 Byte Transponderdaten ausgelesen werden (32 Blöcke a 4 Byte). Gültiger Wertebereich: [1..32]
W#16#0013	Zu lesender Inhalt > 128 Byte	Pro Aktionsaufruf können maximal 128 Byte Daten gelesen werden Um mehr als 128 Byte Daten auszulesen, muss die RD_TAG Aktion mehrmals hintereinander ausgeführt werden.
W#16#0014	WriteTag.iStartBlock < 0	Ungültiger Parameter. Gültiger Wertebereich: [0.. Max Anzahl Transponder Blöcke]
W#16#0015	32 < WriteTag. iNumBlocks <= 0	Pro Aktionsaufruf können maximal 128 Byte Transponderdaten geschrieben werden (32 Blöcke a 4 Byte). Gültiger Wertebereich: [1..32]
W#16#0016	WriteTag.iBlockSize <> 4,8,12,16,…	Ungültige Blockgröße. Gültiger Wertebereich: [4,8,12,16,…]
W#16#0017	Zu schreibender Inhalt > 128 Byte	Pro Aktionsaufruf können maximal 128 Byte Daten geschrieben werden. Um mehr als 128 Byte Daten zu schreiben, muss die WR_TAG Aktion mehrmals hintereinander ausgeführt werden.

Fehlercode	Kurzbeschreibung	Beschreibung
W#16#0018	iLockBlock < 0	Ungültiger iLockBlock Parameter Gültiger Wertebereich: [0.. Max Anzahl Transponder Blöcke]
W#16#**XX**19	Transponderfehler	Es ist ein Transponderfehler bzw. ein Gerätefehler aufgetreten. **XX** = Transponder- / Gerätefehler <u>Transponderfehler:</u> 16#00: No error 16#01: Command not supported 16#02: Command nor recognized 16#03: Option not supported 16#0F: Unknown error 16#10: Block not available 16#11: Block already locked 16#13: Block write error 16#14: Block lock error <u>Gerätefehler:</u> 16#1E: Unknown error 16#1F: CRC error 16#20: Parity error 16#21: Timeout error 16#22: No response error 16#23: Collision error 16#24: Content check error 16#25: Framing error 16#26: Verify error 16#27: Transmit error 16#28: Receive error 16#29: Non addressed error 16#2A: Tag type selection error 16#2B: Max block count error 16#2C: Block length mismatch error 16#46: Slot detect warning Für eine detaillierte Fehlerbeschreibung siehe Gerätebeschreibung.
W#16#001A	Kein Tag im Feld	Es befindet sich kein Tag im Empfangsbereich des RFHs.
W#16#001B	Mehr als ein Tag im Feld	Es befindet sich mehr als ein Tag im Empfangsbereich des RFHs. Dieser Fehler kann nur im Mode 1 auftreten.

Tabelle 7: Fehlercodes

7 Beispiele

Abbildung 9 zeigt eine Beispielbeschaltung des RFH6XX FBs. Die logische Eingangs- und Ausgangsadresse fängt bei Byte 258 (W#16#102) an. Die Länge der in der Hardwarekonfiguration projektierten Module beträgt 32 Bytes. Da der RFH nicht in einem CAN-Netzwerk betrieben wird, wird als CAN-ID fest eine null eingetragen.

Programmaufruf:

```
Network 2 : CALL SICK RFH PNDP FUNCTION BLOCK
Comment:

      CALL  "SICK RFH6XX PNDP" , "INSTANCE_FB73"     FB73 / DB173
       IN_ADDR      :=W#16#102
       IN_LEN       :=32
       OUT_ADDR     :=W#16#102
       OUT_LEN      :=32
       CAN_ID       :="iCanID"                       MW16
       TOUT         :=T#5S
       START_REQ    :="bRequest"                     M10.0
       TRIG_ON      :="bTriggerOn"                   M12.1
       TRIG_OFF     :="bTriggerOff"                  M12.2
       RD_TAG       :="bRdTag"                       M12.3
       WR_TAG       :="bWrTag"                       M12.5
       INVENTORY    :="bInventory"                   M12.7
       LOCK_BLOCK   :="bLockBlock"                   M13.0
       STAY_QUIET   :="bStayQuit"                    M13.1
       COM_TEST     :="bComTest"                     M12.4
       FREE_COMMAND:="bFreeCommand"                  M12.6
       RESET        :="bReset"                       M12.0
       DATA         :="SICK RFH DATA"                DB73
       RD_DONE      :="bRdDone"                      M10.1
       REQ_DONE     :="bReqDone"                     M10.2
       REQ_BUSY     :="bReqBusy"                     M10.3
       ERROR        :="bError"                       M10.4
       ERRORCODE    :="nErrorcode"                   MW14
```

Abbildung 9: Beispielbeschaltung des SICK RFH6XX PNDP FBs

Slot	DP ID	...	Order Number / Designation	I Address	Q Address
1	208		Ctrl Bits in	256...257	
2	224		Ctrl Bits out		256...257
3	64		32 byte input con (0x40,0x9F)	258...289	
4	128		32 byte output con (0x80,0x9F)		258...289

Abbildung 10: Step7 Hardwareprojektierung

7.1 Auslesen von Tag-Inhalten

Zunächst muss bestimmt werden, mit welchem Transponder kommuniziert werden soll. Ist das Bit „Mode.bMode = FALSE“ wird mit dem Transponder kommuniziert, welcher sich aktuell im Lesebereich des RFID Sensors befindet.

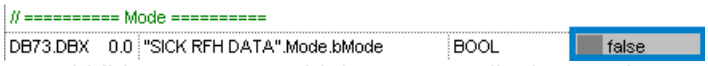

Abbildung 11: Auswahl des Kommunikationsmodus

Anschließend muss definiert werden, welche Inhalte aus dem Transponder ausgelesen werden sollen.

Start Block: 0
Anzahl Blocks: 2 (Anzahl der zu lesenden Blöcke)

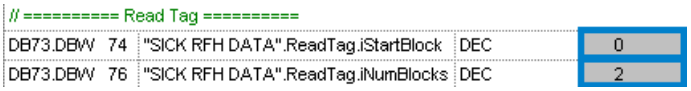

Abbildung 12: Read Block Parameter

Die Leseaktion (bRdTag) wird ausgeführt, sobald das Bit „bRequest“ mit einer positiven Flanke angetriggert wird.

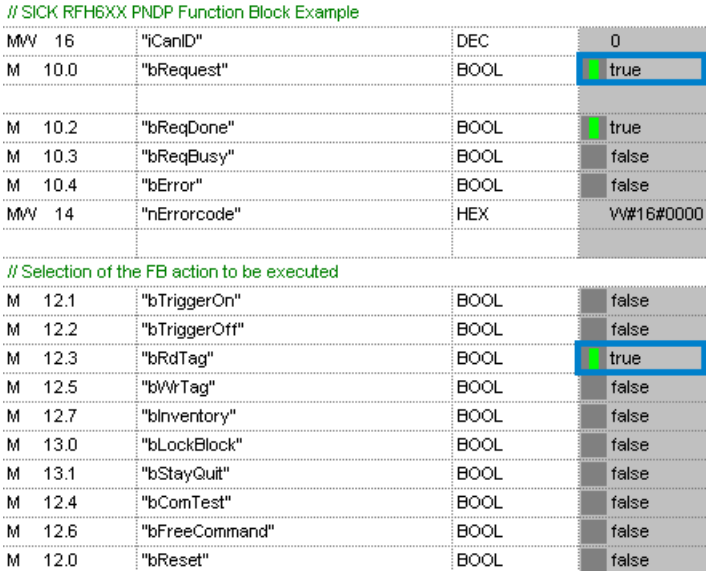

Abbildung 13: Starten der Bausteinfunktion

Die Leseaktion ist abgeschlossen sobald das Bit „bReqDone = TRUE“ signalisiert. Die gelesenen Tag-Inhalte stehen im Array „ReadTag.arrData“ des Nutzdatenbausteins zur Verfügung. Die Variable „ReadTag.iDataLength“ gibt an, wie viele Bytes empfangen wurden bzw. gültig sind.

// ========== Read Tag ==========				
DB73.DBW	74	"SICK RFH DATA".ReadTag.iStartBlock	DEC	0
DB73.DBW	76	"SICK RFH DATA".ReadTag.iNumBlocks	DEC	2
DB73.DBW	78	"SICK RFH DATA".ReadTag.iDataLength	DEC	8
DB73.DBB	80	"SICK RFH DATA".ReadTag.arrData[1]	CHARACTER	'S'
DB73.DBB	81	"SICK RFH DATA".ReadTag.arrData[2]	CHARACTER	'I'
DB73.DBB	82	"SICK RFH DATA".ReadTag.arrData[3]	CHARACTER	'C'
DB73.DBB	83	"SICK RFH DATA".ReadTag.arrData[4]	CHARACTER	'K'
DB73.DBB	84	"SICK RFH DATA".ReadTag.arrData[5]	CHARACTER	' '
DB73.DBB	85	"SICK RFH DATA".ReadTag.arrData[6]	CHARACTER	'A'
DB73.DBB	86	"SICK RFH DATA".ReadTag.arrData[7]	CHARACTER	'G'
DB73.DBB	87	"SICK RFH DATA".ReadTag.arrData[8]	CHARACTER	' '
DB73.DBB	88	"SICK RFH DATA".ReadTag.arrData[9]	CHARACTER	B#16#00
DB73.DBB	89	"SICK RFH DATA".ReadTag.arrData[10]	CHARACTER	B#16#00

Abbildung 14: Gelesene Tag-Inhalte

7.2 Schreiben von Tag-Inhalten

Zunächst muss bestimmt werden, mit welchem Transponder kommuniziert werden soll. Ist das Bit „Mode.bMode = TRUE“ wird mit einem vorgegebenen Transponder kommuniziert, dessen UID im Vorfeld bekannt sein muss (hier: E0 04 01 00 06 D2 37 45).

// ========== Mode ==========				
DB73.DBX	0.0	"SICK RFH DATA".Mode.bMode	BOOL	true
DB73.DBB	2	"SICK RFH DATA".Mode.arrUID[1]	HEX	B#16#E0
DB73.DBB	3	"SICK RFH DATA".Mode.arrUID[2]	HEX	B#16#04
DB73.DBB	4	"SICK RFH DATA".Mode.arrUID[3]	HEX	B#16#01
DB73.DBB	5	"SICK RFH DATA".Mode.arrUID[4]	HEX	B#16#00
DB73.DBB	6	"SICK RFH DATA".Mode.arrUID[5]	HEX	B#16#06
DB73.DBB	7	"SICK RFH DATA".Mode.arrUID[6]	HEX	B#16#D2
DB73.DBB	8	"SICK RFH DATA".Mode.arrUID[7]	HEX	B#16#37
DB73.DBB	9	"SICK RFH DATA".Mode.arrUID[8]	HEX	B#16#45

Abbildung 15: Vorgabe der Transponder UII

Anschließend muss definiert werden, welche Inhalte auf den Tag geschrieben werden und wo diese abgelegt werden sollen.

Start Block: 0
Anzahl der Blocks: 3 (Anzahl der zu schreibenden Blöcke)
Blockgröße: 4 (Transponderabhängig)
Daten: 'Hello World '

// ========== Write Tag ==========			
DB73.DBW 208	"SICK RFH DATA".WriteTag.iStartBlock	DEC	0
DB73.DBW 210	"SICK RFH DATA".WriteTag.iNumBlocks	DEC	3
DB73.DBW 212	"SICK RFH DATA".WriteTag.iBlockSize	DEC	4
DB73.DBB 214	"SICK RFH DATA".WriteTag.arrData[1]	CHARACTER	'H'
DB73.DBB 215	"SICK RFH DATA".WriteTag.arrData[2]	CHARACTER	'e'
DB73.DBB 216	"SICK RFH DATA".WriteTag.arrData[3]	CHARACTER	'l'
DB73.DBB 217	"SICK RFH DATA".WriteTag.arrData[4]	CHARACTER	'l'
DB73.DBB 218	"SICK RFH DATA".WriteTag.arrData[5]	CHARACTER	'o'
DB73.DBB 219	"SICK RFH DATA".WriteTag.arrData[6]	CHARACTER	' '
DB73.DBB 220	"SICK RFH DATA".WriteTag.arrData[7]	CHARACTER	'W'
DB73.DBB 221	"SICK RFH DATA".WriteTag.arrData[8]	CHARACTER	'o'
DB73.DBB 222	"SICK RFH DATA".WriteTag.arrData[9]	CHARACTER	'r'
DB73.DBB 223	"SICK RFH DATA".WriteTag.arrData[10]	CHARACTER	'l'
DB73.DBB 224	"SICK RFH DATA".WriteTag.arrData[11]	CHARACTER	'd'
DB73.DBB 225	"SICK RFH DATA".WriteTag.arrData[12]	CHARACTER	' '

Abbildung 16: Write Block Parameter

Die Schreibaktion (bWrTag) wird ausgeführt, sobald das Bit „bRequest“ mit einer positiven Flanke angetriggert wird.

// SICK RFH6XX PNDP Function Block Example			
MW 16	"iCanID"	DEC	0
M 10.0	"bRequest"	BOOL	true
M 10.2	"bReqDone"	BOOL	true
M 10.3	"bReqBusy"	BOOL	false
M 10.4	"bError"	BOOL	false
MW 14	"nErrorcode"	HEX	W#16#0000
// Selection of the FB action to be executed			
M 12.1	"bTriggerOn"	BOOL	false
M 12.2	"bTriggerOff"	BOOL	false
M 12.3	"bRdTag"	BOOL	false
M 12.5	"bWrTag"	BOOL	true
M 12.7	"bInventory"	BOOL	false
M 13.0	"bLockBlock"	BOOL	false
M 13.1	"bStayQuit"	BOOL	false
M 12.4	"bComTest"	BOOL	false
M 12.6	"bFreeCommand"	BOOL	false
M 12.0	"bReset"	BOOL	false

Abbildung 17: Starten der Bausteinfunktion

Die Schreibaktion ist abgeschlossen sobald das Bit „bReqDone = TRUE“ signalisiert.