Technische Information

SICK
RFH6xx AOI

SICK_RFH_EIP Add-On Instruction für
Rockwell / Allen Bradley Logix5000 Steuerungen

SICK
Sensor Intelligence.

Versionshistorie

Version	Datum	Beschreibung
V1.0	04.01.2013	Initiale Version
V1.1	07.11.2013	Fix arithmetik error alle 32768 eingehende Leseergebnisse
V1.2	10.07.2017	Fix: Empfang von Leseergebnissen via Feldbustrigger

Inhaltsverzeichnis

1 Zu diesem Dokument

Bitte lesen Sie dieses Kapitel sorgfältig, bevor Sie mit dieser Anleitung und der SICK_RFH_EIP AOI arbeiten.

1.1 Funktion dieses Dokuments

Diese Anleitung beschreibt den Umgang mit der SICK_RFH_EIP Add-On Instruction. 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

Die Add-On Instruction (AOI) wird zur Kommunikation zwischen einer Rockwell Steuerung und einem SICK RFH6xx RFID Interrogator verwendet. Das Gerät muss hierfür in das EtherNet/IP Umfeld der Steuerung eingebunden werden. Die Kommunikation erfolgt über die zyklischen Prozessdaten (implizite Kommunikation).

Die folgende Abbildung zeigt die Darstellung der AOI in der Function Block Diagram (FBD) Ansicht.

Abbildung 1: Darstellung der SICK_RFH_EIP AOI

Bausteinfunktionalitäten:

- Senden eines Triggerbefehls (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 (Anzeigen aller Transponder 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 Einbinden der AOI in RSLogix5000

Die AOI kann mit allen Rockwell Steuerungen verwendet werden, die mit RSLogix5000 V16 oder höher programmiert werden können.

Die Implementierung des SICK_RFH_EIP Bausteins wird über eine Add-On Instruction (AOI) gehandhabt. Die AOI beinhaltet eine Programmroutine, die an einer beliebigen Stelle im Anwenderprogramm zyklisch aufgerufen werden muss.

3.1 SOPAS Gerätekonfiguration

Um den EtherNet/IP Bus im SICK Gerät zu aktivieren, muss in SOPAS-ET unter dem Menüpunkt *Network / Interfaces /IOs → Ethernet → EtherNet/IP* die folgenden Einstellungen vorgenommen werden:

- Enable EtherNet/IP at boot-up of device: Aktivieren
- Communication Mode: with Handshake

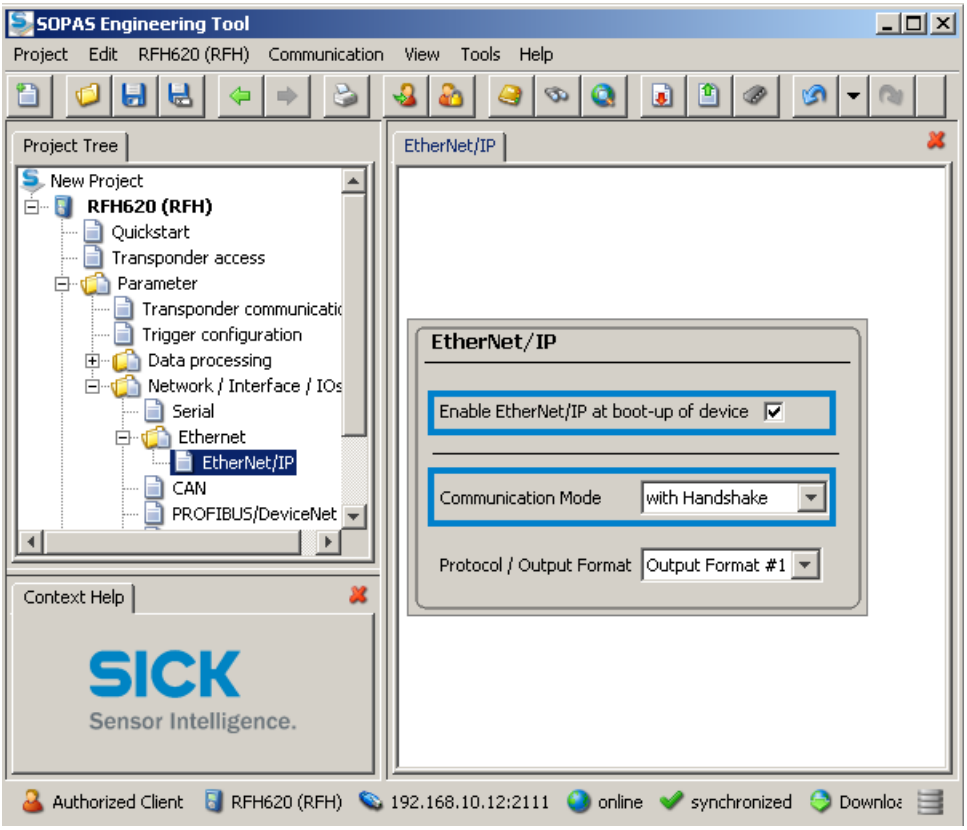


Abbildung 2: Aktivieren der EtherNet/IP Kommunikation in SOPAS-ET

Die AOI kommuniziert über die zyklischen Prozessdaten mit dem RFH6xx (Implizite EtherNet/IP Kommunikation). Die Input-Assembly und die Output-Assembly beinhalten die Prozessdaten des Sensors. Die Länge der Assemblies gibt an, wie viele Daten in einem Buszyklus übertragen werden können. Beim RFH ist die Größe der Assemblies mit 200 Byte fest vordefiniert. Sollte der Inhalt des Leseergebnisses 200 Byte überschreiten, wird das Telegram fragmentiert übertragen. Die einzelnen Fragmente werden anschließend von der AOI wieder zusammengesetzt.

3.2 Hardwarekonfiguration

Um mit RSLogix5000 auf die Input- / Output- Assemblies des RFHs zugreifen zu können, muss zunächst der verwendete Sensor projektiert werden.

Klicken Sie mit der rechten Maustaste auf das Symbol *Ethernet* und wählen Sie die Auswahl *New Module…*.

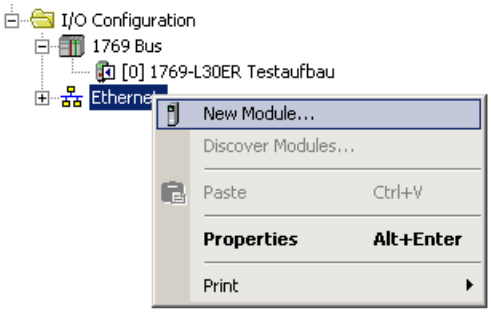


Abbildung 3: Neues Ethernetmodul in RSLogix5000 einfügen

Wählen Sie im Dialog *Select Module* das Modul *ETHERNET-MODULE (Generic Ethernet Module)* aus und klicken Sie anschließend auf *Create* um das Modul in die Hardwarekonfiguration aufzunehmen.

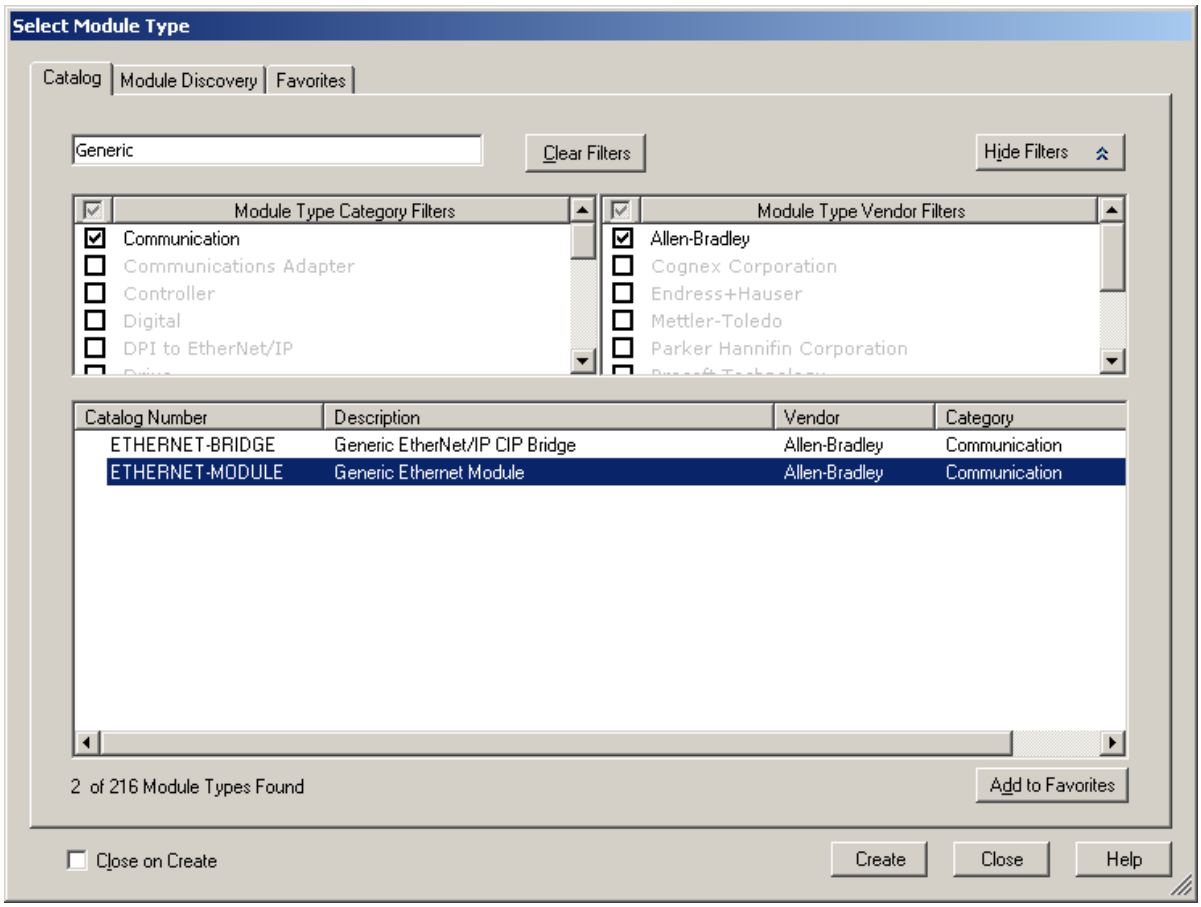


Abbildung 4: Auswahl des Generic Modules in RSLogix5000

Geben Sie im Dialog *New Module* die Einstellungen für *Input*, *Output*, sowie *Configuration* ein.

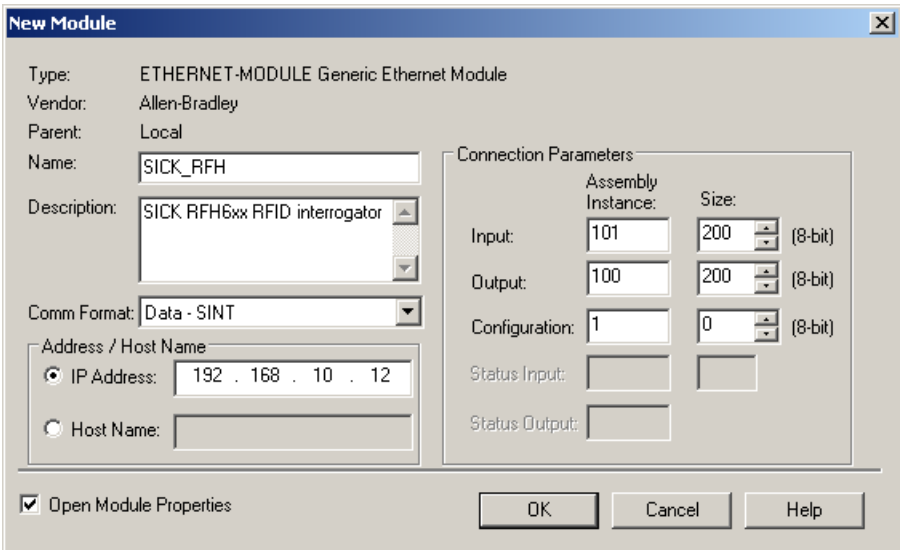


Abbildung 5: Assembly Einstellungen des SICK Sensors

Beispiel:

- Name: SICK_RFH (Name ist frei wählbar)
- Comm Format: Data – SINT
- IP Address: 192.168.10.12 (IP-Adresse des SICK Sensors)
- Input Assembly Instance: 101
- Input Assembly Size: 200
- Output Assembly Instance: 100
- Input Assembly Size: 200
- Configuration Assembly Instance: 1
- Configuration Assembly Size: 0 (Keine Konfigurations-Assembly vorhanden)

Laden Sie die Konfiguration wie folgt in die Steuerung.

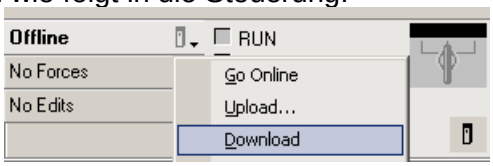


Abbildung 6: Download der SPS Konfiguration

Die Statusanzeigen (Run Mode, Controller OK und I/O) signalisieren, ob die Verbindung zum Sensor erfolgreich hergestellt wurde.

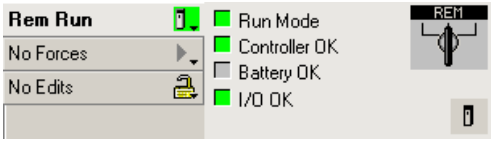


Abbildung 7: Kontrolle der Kommunikation

3.3 AOI Import

Um die SICK_RFH_EIP AOI im Anwenderprogramm zu verwenden, muss diese zunächst über *File → Import Component → Add-On Instruction…* in ein bestehendes Projekt importiert werden.

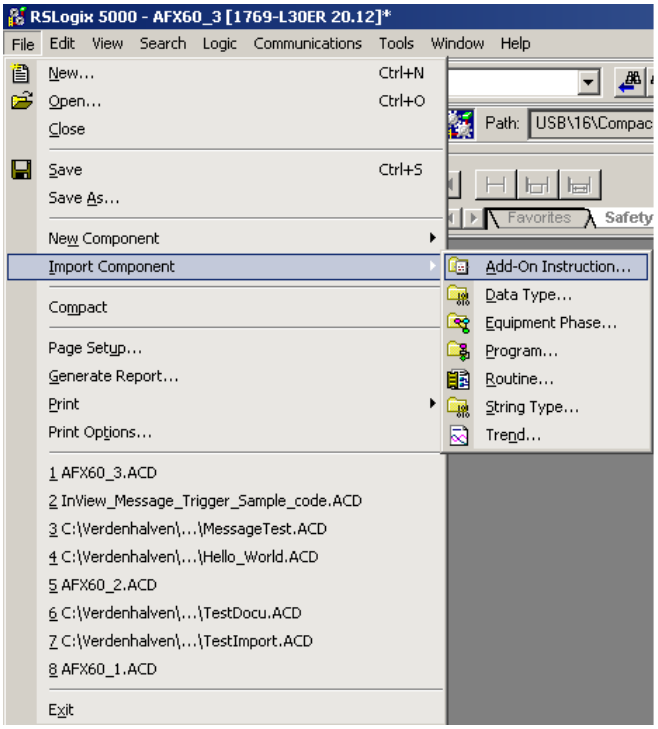


Abbildung 8: Import der SICK_RFH_EIP Add-On Instruction

4 Bausteinbeschreibung

Die Add-On Instruction (AOI) ist eine asynchron arbeitende Routine, d.h. die Bearbeitung erstreckt sich über mehrere Aufrufe. Dies setzt voraus, dass die Routine zyklisch im Anwenderprogramm aufgerufen wird.

Der Baustein kapselt die AOI „SICK_CCOM_CLV_RFH_EIP“, die die Kommunikation zwischen SPS und Sensor abarbeitet. Die AOI „SICK_COLA_ACCESS“ wird intern verwendet, um die Rohdaten des RFHs zu interpretieren.

4.1 Bausteinspezifikationen

Bausteinname:	SICK_RFH_EIP
Version:	1.2
Aufgerufene Bausteine:	SICK_CCOM_CLV_RFH_EIP SICK_COLA_ACCESS
Verwendete UDTs:	SICK_RFH_Data └ SICK_RFH_Mode └ SICK_RFH_Inventory └ SICK_RFH_ReadTag └ SICK_RFH_WriteTag └ SICK_RFH_TagInfo └ SICK_FreeCommand └ SICK_ReadingResult
Bausteinaufruf:	Zyklisch
Erstellsprache:	Strukturierter Text (ST)
RSLogix5000 Version:	RSLogix5000 V20.01.00 (CPR 9 SR 5)

4.2 Arbeitsweise

Um die SICK_RFH_EIP Routine einsetzen zu können, müssen zunächst die folgenden Bausteinparameter beschaltet werden:

arrInputAssembly: Verweiß auf das Input Assembly Array, welches automatisch bei der Geräteprojektierung in den Controller Tags angelegt wird.

arrOutputAssembly: Verweiß auf das Output Assembly Array, welches automatisch bei der Geräteprojektierung in den Controller Tags angelegt wird.

stData: Die zur Routine gehörende Datenstruktur (UDT) *SICK_RFH_Data* beinhaltet Ein- und Ausgabeparameter der unterstützten Bausteinaktionen. Die UDT muss instanziiert und dem Eingangsparameter „stData“ übergeben werden.

Ausführbare Bausteinfunktionen:

- Trigger on → Öffnet das Lesetor des Gerätes über ein Kommando
- Trigger off → Schließt das Lesetor des Gerätes über ein 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 sich im Lesefeld befindenden RFID-Tags
- Kommunikationstest → Prüft, ob das Gerät per „sRI0“ (Kommando für eine Geräteidentifikation) erreichbar ist
- Save Permanent → Speichert alle Geräteparameter dauerhaft im Gerät ab
- Free Command → Ausführen eines frei wählbaren CoLa Kommandos

Um eine Bausteinaktion (*bTriggerOn*, *bTriggerOff*, 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 *bRequest* mit einer positiven Flanke (Signalwechsel von logisch null auf eins) angetriggert werden. Solange noch keine gültige Geräteantwort empfangen wurde, wird dies über den Parameter *bReqBusy* signalisiert.

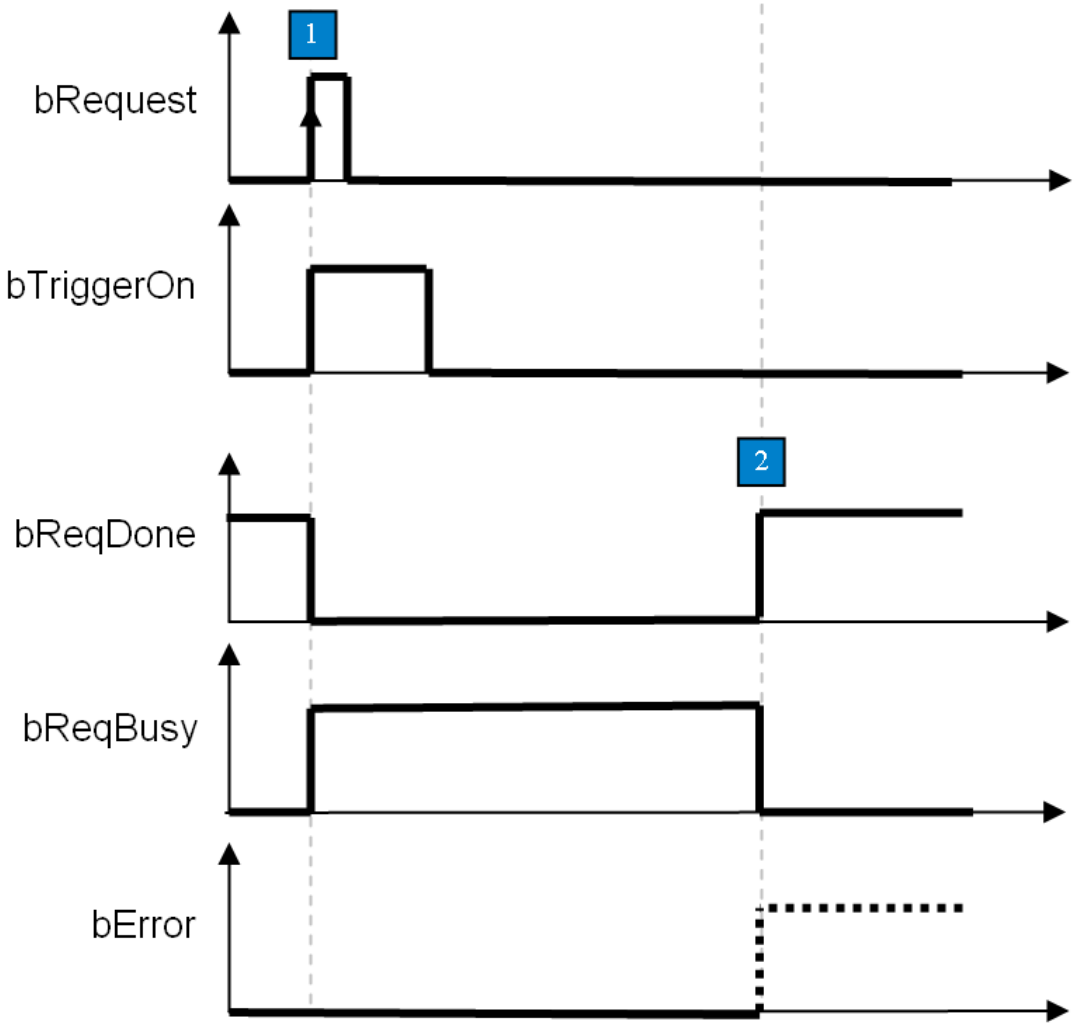
Wenn der Baustein am Ausgangsparameter *bReqDone* = *TRUE* signalisiert, wurde die Aktion erfolgreich durchgeführt. Wurden bei dieser Aktion (z.B. *bFreeCommand*) Daten vom Gerät angefordert, werden diese in den jeweiligen Datenbereich des instanziierten UDTs (*stData*) kopiert.

Daten die per Triggerbefehl (*bTriggerOn*, *bTriggerOff*) oder direkt vom Gerät gesendet werden (z.B. direkter Trigger über eine Lichtschranke), werden in der Datenstruktur (*ReadingResult.sResult*) abgelegt. Der Ausgangsparameter *bRdDone* zeigt für einen SPS Zyklus an, dass neue Daten empfangen wurden. Die vom Gerät gesendeten Daten können im SOPAS Ausgabeformat geändert, bzw. angepasst werden (siehe Kapitel 4.6).

4.3 Verhalten im Fehlerfall

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

4.4 Timing



1: Anforderung durch Pos Flanke an bRequest
Die gewünschte Aktion (hier *bTriggerOn*) muss vorher/zeitgleich ausgewählt werden. Es darf nur eine Aktion zeitgleich ausgewählt werden, sonst wird mit *bError* = *TRUE* abgebrochen.

2: Wenn alle Kommandos gesendet sind und alle Antworten empfangen wurden, wird die Aktion mit *bReqDone* = *TRUE* beendet. Wenn die Aktion fehlerhaft verläuft, wird mit *bError* = *TRUE* beendet. Bei Abbruch mit *bError* enthält *iErrorcode* den aufgetretenen Fehler.

4.5 Werteübergabe

Die mitgelieferte UDT „SICK_RFH_Data“ beinhaltet Ein- und Ausgabeparameter aller unterstützten Bausteinaktionen. Die Datenstruktur ist fest vordefiniert und darf, bis auf den letzten Eintrag (ReadingResult.sResult), nicht geändert werden (siehe Kapitel 4.7: Empfangen von Leseergebnissen > 200 Byte.

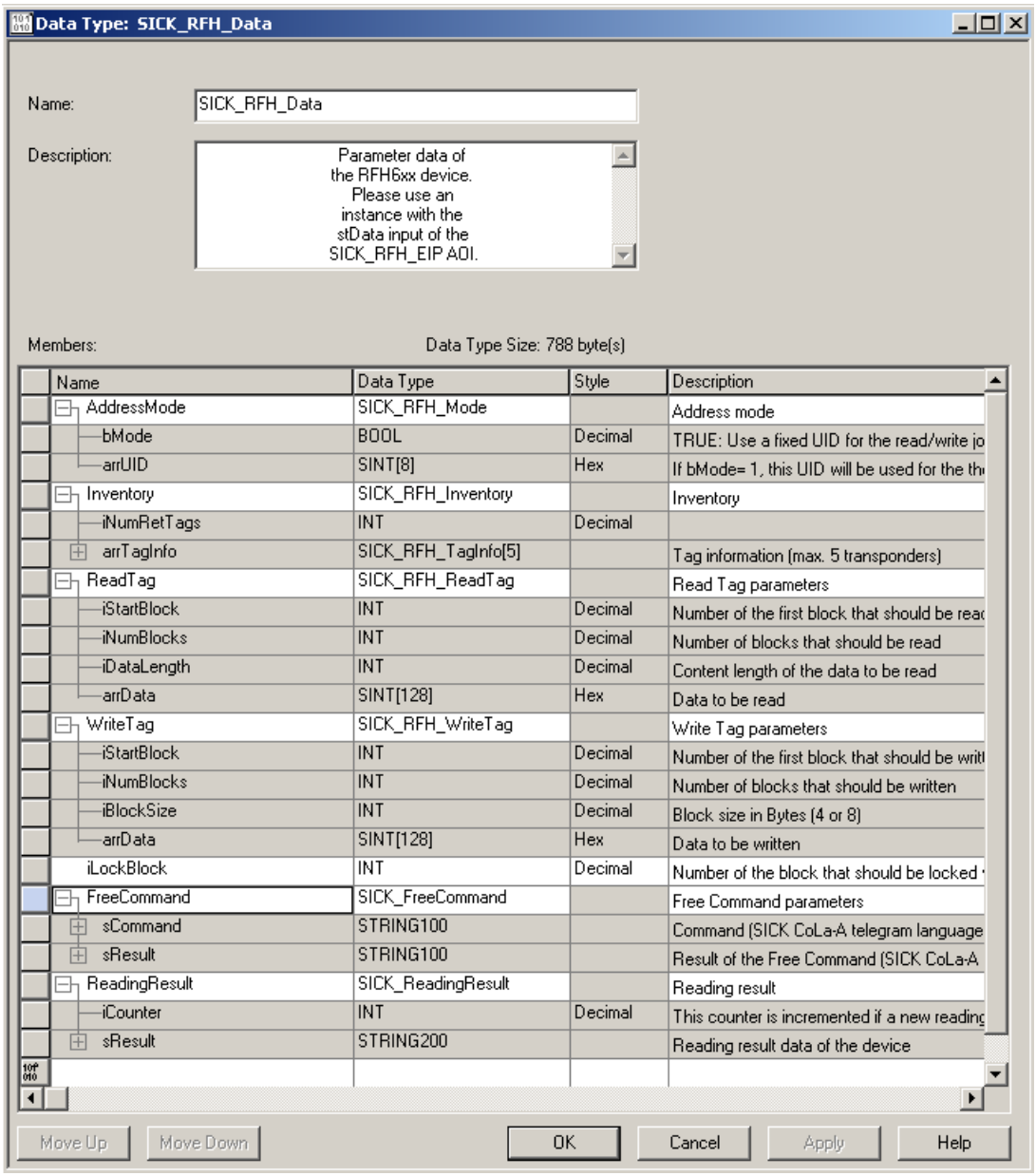


Abbildung 9: Datenstruktur des SICK_RFH_Data UDTs

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 wird die UID (Unique Identifier) verwendet.

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
AddressMode. bMode	Input	BOOL	Adressierungsmodus FALSE: Mode 1 aktiv TRUE: Mode 2 aktiv
AddressMode. arrUID	Input/Output	Array [0..7] OF SINT	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 RFHs nach aktiven Transpondern. Für jeden erkannten Transponder (max. 5 Transponder) stellt die AOI die folgenden Informationen zur Verfügung.

Parameter	Deklaration	Datentyp	Beschreibung
Inventory. iNumRetTags	Output	INT	Anzahl der erkannten Transponder
Inventory. arrTagInfo[ ].iError	Output	INT	Transponder Errorcode (siehe RFH Betriebsanleitung)
Inventory. arrTagInfo[ ].iRSSI	Output	INT	RSSI (Signalstärke des erkannten Transponders)
Inventory. arrTagInfo[ ].iDSFID	Output	INT	DSFID des erkannten Transponders
Inventory. arrTagInfo[ ].arrUID	Output	ARRAY [0..7] OF SINT	UID des erkannten Transponders 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 jeden Lesevorgang muss angegeben werden, welche Datenblöcke des Transponders ausgelesen werden sollen.

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 Gültiger Wertebereich: [1..32]
ReadTag. iDataLength	Output	INT	Länge des gelesenen Inhalts in Byte
ReadTag.arrData	Output	ARRAY [0..127] OF SINT	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. Gültiger Wertebereich: [1..32]
WriteTag. iBlockSize	Input	INT	Bytegröße eines Blocks Gültiger Wertebereich: [4,8]
WriteTag.arrData	Input	ARRAY [0..127] OF SINT	Daten die in die Datenblöcke des Transponders 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 Gerät zu kommunizieren. Hierfür ist es erforderlich, das Kommando in den Parameter *sCommand* der Struktur *FreeCommand* zu hinterlegen. Die Kommandos können der Gerätebeschreibung oder SOPAS-ET 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 4: Free Command Parameter

4.5.7 Reading Result

In dem Datenstring *ReadingResult.sResult* werden Daten abgelegt, die per Triggerbefehl (*bTriggerOn*, *bTriggerOff*) oder direkt vom Gerät gesendet werden (z.B. direkter Trigger über eine Lichtschranke oder Feldbus). Der Ausgangsparameter *bRdDone* signalisiert, ob Daten empfangen wurden.

Parameter	Deklaration	Datentyp	Beschreibung
ReadingResult. iCounter	Output	INT	Der Empfangszähler wird um eins inkrementiert, sobald ein neues Leseergebnis empfangen wurde.Der Empfangszähler wird automatisch auf null zurückgesetzt, sobald der Wert 32767 überschritten wird. Wertebereich: [0..32767]
ReadingResult. sResult	Output	STRING 200	Empfangende Antwort auf ein Triggersignal (Über das SOPAS Ausgabeformat definierbar). Die maximale Länge der empfangenden Daten beträgt 200 Bytes. Kapitel 4.7 beschreibt das Vorgehen beim Empfang von längeren Datentelegrammen.

Tabelle 5: Reading Result Parameter

4.6 Triggereinstellungen

Sobald der RFH angetriggert wird, wird ein benutzerdefiniertes Telegram vom Gerät gesendet wird. Dieses Telegram kann im Ausgabeformat unter SOPAS-ET frei konfiguriert werden.

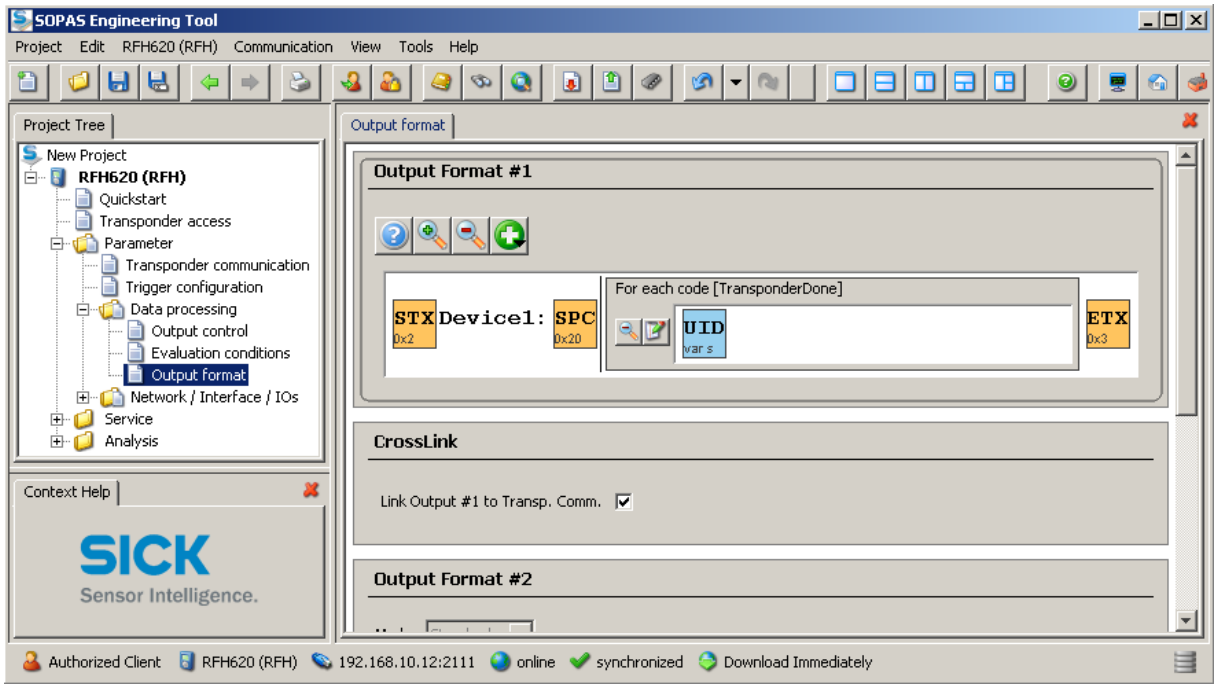


Abbildung 10: Beispielkonfiguration des Ausgabeformats in SOPAS-ET

Der RFH kann auf verschiedene Arten getriggert werden.

- Softwaretrigger über die AOI (*bTriggerOn* / *bTriggerOff*)
- Feldbustrigger über die AOI (*arrControl*)
- Hardwaretrigger über den Sensor1 Eingang des RFHs
- Auto Trigger

Wenn ein Triggerergebnis (Leseergebnis) vom Baustein empfangen wird, wird dies immer über den Ausgangsparameter „bRdDone“ signalisiert.

4.6.1 Trigger über Kommando

Damit eine Triggerung über die SPS erfolgen kann, muss die Triggerquelle zuvor über SOPAS-ET auf „Kommando“ bzw. auf „Fieldbus“ eingestellt werden. Abbildung 11 zeigt, wie unter dem Menüpunkt „Trigger configuration“ der RFH parametriert wird.

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

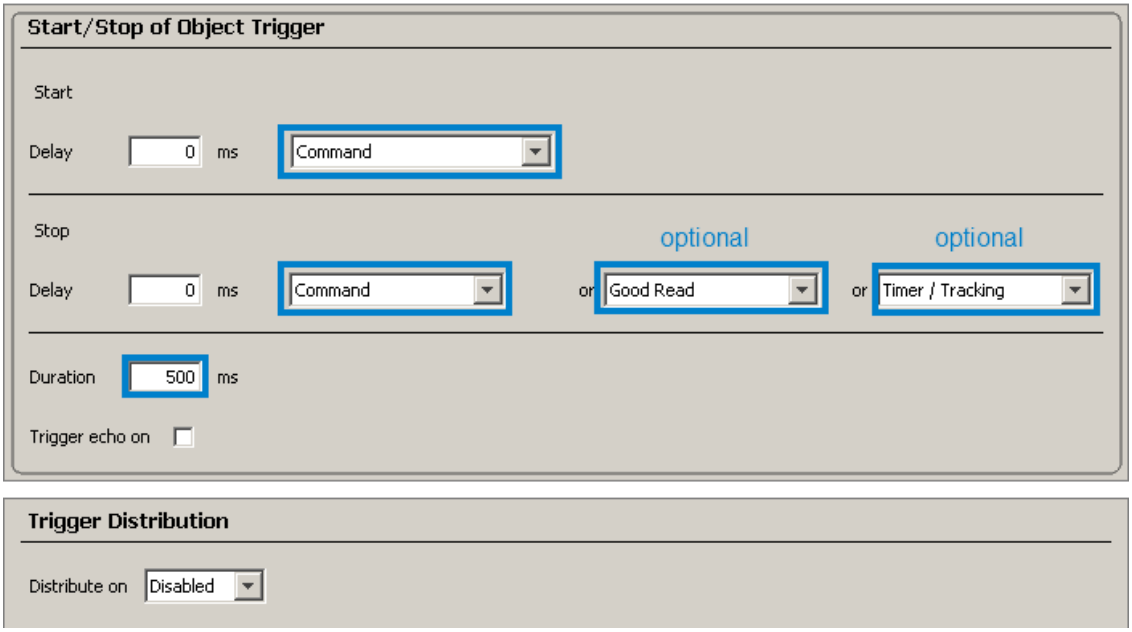


Abbildung 11: SOPAS Trigger Einstellungen

4.6.2 Feldbus Trigger

Um das Gerät direkt über den Feldbus zu triggern, muss die Triggerquelle in SOPAS-ET auf „Fieldbus“ eingestellt werden. Anschließend kann der RFH getriggert werden, indem das erste Bit im arrControl Array (arrControl[0].0) gesetzt wird.

- Start, wenn arrControl[0].0 = TRUE
- Stop, wenn arrControl[0].0 = FALSE. 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.

4.7 Empfangen von Leseergebnissen > 200 Byte

Die AOI ist darauf ausgelegt, Leseergebnisse bis zu einer Länge von 200 Bytes zu empfangen. Sollen längere Daten empfangen werden, muss die Routine an der folgenden Stelle abgeändert werden:

<u>Änderung in der SICK_RFH_Data UDT:</u>
In der mitgelieferten UDT (SICK_RFH_Data) muss die Länge des Strings „ReadingResult.sResult“ so angepasst werden, dass das zu empfangende Leseergebnis in den Datenbereich der Variablen passt.

ReadingResult	SICK_ReadingResult		Reading result
iCounter	INT	Decimal	This counter is incremented if a new reading result has arrived
sResult	STRING200		Reading result data of the device

Abbildung 12: Empfangen von Leseergebnissen > 200 Bytes (Änderung in der UDT)

Die maximale Stringlänge ist auf 500 Zeichen begrenzt.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
arrInput Assembly	IN/OUT	SINT[1]	Verweiß auf das Input Assembly Array, welches automatisch bei der Geräteprojektierung in den Controller Tags angelegt wird. Die Assembly muss eine Größe von 200Byte haben. Beispiel: arrInputAssembly:= myRFH:I.Data
arrOutput Assembly	IN/OUT	SINT[1]	Verweiß auf das Output Assembly Array, welches automatisch bei der Geräteprojektierung in den Controller Tags angelegt wird. Die Assembly muss eine Größe von 200Byte haben. Beispiel: arrOutputAssembly:= myRFH:O.Data
arrControl	IN/OUT	SINT[3]	Control Array zum triggern des Sensors über den Feldbus. arrControl[0] = Control Byte 1 arrControl[1] = Control Byte 2 arrControl[2] = Status Byte des CM-Protokolls Beispiel: Zum Triggern des Sensors über den Feldbus, muss das Bit arrControl[0].0 gesetzt werden. Voraussetzung hierfür ist, dass die Triggerquelle in SOPAS auf „Feldbustrigger“ eingestellt ist. Die Definition der Control-Bits im Array können in der der Betriebanleitung entnommen werden.
iTimeout	INPUT	DINT	Zeit [ms], nachdem ein Timeout-Fehler ausgelöst wird.
iCanID	INPUT	INT	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.
bRequest	INPUT	BOOL	Positive Flanke: Ausführen der gewählten Bausteinaktion.
bTriggerOn	INPUT	BOOL	Bausteinaktion: Ausführen eines Geräte Triggers (Triggerfenster öffnen)
bTriggerOff	INPUT	BOOL	Bausteinaktion: Ausführen eines Geräte Triggers (Triggerfenster schließen) Das vom Gerät gesendet Ergebnis (SOPAS Ausgabeformat) wird in der Variablen „ReadingResult.sResult“ der übergebenden Datenstruktur (SICK_RFH_Data) abgelegt.

Parameter	Deklaration	Datentyp	Beschreibung
bReadTag	INPUT	BOOL	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).
bWriteTag	INPUT	BOOL	Bausteinaktion: Schreiben von Tag Inhalten. Die Aktion setzt voraus, dass die Parameter der Struktur „WriteTag“ des übergebenen UDTs (SICK_RFH_Data) 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).
bInventory	INPUT	BOOL	Sucht im Empfangsbereich nach aktiven Transpondern und gibt deren UID, DSFID und die RSSI Signalstärke zurück.
bLockBlock	INPUT	BOOL	Schützt einen definierten Block gegen Wiederbeschreiben. Die Aktion setzt voraus, dass der Parameter iLockBlock im übergebenen UDT mit einer gültigen Blocknummer belegt ist (siehe Kapitel 4.5.2). Die Aktion sperrt den gewählten Block permanent. Ein entsperren ist nicht möglich.
bStayQuiet	INPUT	BOOL	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).
bComTest	INPUT	BOOL	Bausteinaktion: Ausführen eines Kommunikationstests. bReqDone= TRUE: Kommunikation OK bReqDone= FALSE: Kommunikation nicht OK

Parameter	Deklaration	Datentyp	Beschreibung
bFree Command	INPUT	BOOL	Bausteinaktion: Ausführen eines freien Kommandos. Die Aktion setzt ein gültiges CoLa-Kommando in der Datenstruktur (FreeCommand. sCommand) voraus (siehe Kapitel 4.5.1). Die Kommandoantwort steht nach einer erfolgreichen Übertragung (bReqDone=TRUE) im Result-String der Datenstruktur zur Verfügung.
stData	IN/OUT	SICK_ RFH_ Data	Übergabe der zugehörigen UDT Struktur (SICK_RFH_Data), die für die Parametrierung der Bausteinfunktionen sowie für das Ablegen des Leseergebnisses benötigt wird.
bRdDone	OUTPUT	BOOL	Positive Flanke: Neues Leseergebnis empfangen. Der Inhalt des Leseergebnissen kann mit SOPAS-ET konfiguriert werden (siehe Kapitel 4.6).
bReqDone	OUTPUT	BOOL	Zeigt an, ob die gewählte Bausteinaktion fehlerfrei durchgeführt wurde. TRUE: Bearbeitung abgeschlossen FALSE: Bearbeitung nicht abgeschlossen
bReqBusy	OUTPUT	BOOL	Auftrag ist in Bearbeitung.
bError	OUTPUT	BOOL	Fehler Bit: 0: Kein Fehler 1: Abbruch mit Fehler
iErrorcode	OUTPUT	DINT	Fehlerstatus (siehe Fehlercodes)

Tabelle 6: Bausteinparameter

5 Fehlercodes

Der Parameter *iErrorcode* enthält die folgenden Fehlerinformationen:

Fehlercode	Kurzbeschreibung	Beschreibung
16#0000_0000	Kein Fehler	Kein Fehler
16#0000_0001	Timeout Fehler	Auftrag konnte innerhalb der gewählten Timeoutzeit nicht ausgeführt werden Dies könnte folgende Ursachen haben: - Gerät ist nicht mit der SPS Verbunden - CAN-Bus Teilnehmer nicht vorhanden - Bearbeitungszeit des Kommandos > Time-out Zeit
16#0000_0002	Interner Bausteinfehler	Interner Bausteinfehler
16#0000_0003	Keine oder mehr als eine Bausteinaktion angewählt	Es kann immer nur eine Bausteinfunktion gleichzeitig ausgeführt werden
16#0000_0004	Reserviert	Reserviert
16#0000_0005	100 < Free Command Länge <=0	Ungültige Länge des freien Kommandos Gültiger Wertebereich: [1…100]
16#0000_0006	Antwort des freien Kommandos > 100 Byte	Die Antwort auf das gesendete freie Kommando ist länger 100 Byte.
16#0000_0007	63 < iCanID < 0	Ungültige CAN-ID Gültiger Wertebereich: [0..63]
16#0000_0008	Reserviert	Reserviert
16#**XXXX**_0009	Kommunikationsfehler	Kommunikation zum Gerät kann nicht hergestellt werden. **XXXX** = Fehlercode des SICK_CCOM_RFH_RFH_EIP Bausteins (siehe Bausteindokumentation).
16#00**XX**_000A	Gerätefehler	Es ist ein Gerätefehler aufgetreten (’sFA XX’) **XX** = Gerätefehler (siehe Gerätedokumentation)

Fehlercode	Kurzbeschreibung	Beschreibung
16#0000_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)
16#0000_000C … 16#0000_000F	Reserviert	Reserviert
16#0000_0010	Tags im Feld > 5 (Inventory)	Inventory kann nicht ausgeführt werden, da sich mehr als 5 Transponder im Lesefeld des RFHs befinden.
16#0000_0011	ReadTag.iStartBlock < 0	Ungültiger Lesebeginn (Read Tag)
16#0000_0012	32 < ReadTag. iNumBlocks <= 0	Pro Aktionsaufruf können maximal 128 Byte Transponderdaten ausgelesen werden (32 Blöcke a´ 4 Byte). Gültiger Wertebereich: [1..32]
16#0000_0013	Zu lesender Inhalt > 128 Byte	Pro Aktionsaufruf können maximal 128 Byte Daten gelesen werden Um mehr als 128 Byte Daten auszulesen, muss die „Read Tag“ Aktion mehrmals hintereinander ausgeführt werden.
16#0000_0014	WriteTag.iStartBlock < 0	Ungültiger Parameter. Gültiger Wertebereich: [0.. Max Anzahl Transponder Blöcke]
16#0000_0015	32 < WriteTag. iNumBlocks <= 0	Pro Aktionsaufruf können maximal 128 Byte Transponderdaten geschrieben werden (32 Blöcke a 4 Byte). Gültiger Wertebereich: [1..32]
16#0000_0016	WriteTag.iBlockSize <> 4,8	Ungültige Blockgröße. Gültiger Wertebereich: [4,8]
16#0000_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 „Write Tag“ Aktion mehrmals hintereinander ausgeführt werden.

Fehlercode	Kurzbeschreibung	Beschreibung
16#0000_0018	iLockBlock < 0	Ungültiger iLockBlock Parameter Gültiger Wertebereich: [0.. Max Anzahl Transponder Blöcke]
16#00**XX**_0019	Transponderfehler	Es ist ein Transponderfehler aufgetreten. **XX** = Transponderfehler 16#00: NO_ERROR 16#01: VICC_CMD_NOT_SUPPORTED 16#02: VICC_CMD_NOT_RECOGNIZED 16#03: VICC_OPTION_NOT_SUPPORTED 16#0F: VICC_UNKNOWN_ERROR 16#10: VICC_BLCK_NOT_AVAILABLE 16#11: VICC_BLCK_ALRDY_LOCKED 16#13: VICC_BLCK_WRITE_ERROR 16#14: VICC_BLCK_LOCK_ERROR 16#1E: VCD_UNKNOWN_ERROR 16#1F: VCD_CRC_ERROR 16#20: VCD_PARITY_ERROR 16#21: VCD_TIMEOUT_ERROR 16#22: VCD_NO_RESP_ERROR Weitere Fehlercodes siehe Gerätebeschreibung.
16#0000_001A	Kein Tag im Feld	Es befindet sich kein Tag im Empfangsbereich des RFHs. Dieser Fehler kann nur im Mode 1 auftreten.
16#0000_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.
sReadResult.LEN = -1	Eingehendes Leseergebnis > arrRecord (500 Byte) Leseergebnis > sReadResult String (200 Byte)	Das eingehende Leseergebnis ist länger als das Record-Array (arrRecord), bzw. größer als der sReadResult String. Die AOI kann Leseergebnisse bis zu einer Größe von 200 Byte empfangen. Zum empfangen von Leserergebnissen größer 200 Byte siehe Kapitel 4.7.

Tabelle 7: Fehlercodes

6 Beispiel

Abbildung 13 zeigt eine Beispielbeschaltung der SICK_RFH_EIP AOI. Da sich das Gerät in keinem CAN-Netzwerk befindet, wird als CAN-ID eine null eingetragen. Die Input Assembly und die Output Assembly des Geräts werden direkt mit der Routine verknüpft.

Programmaufruf:

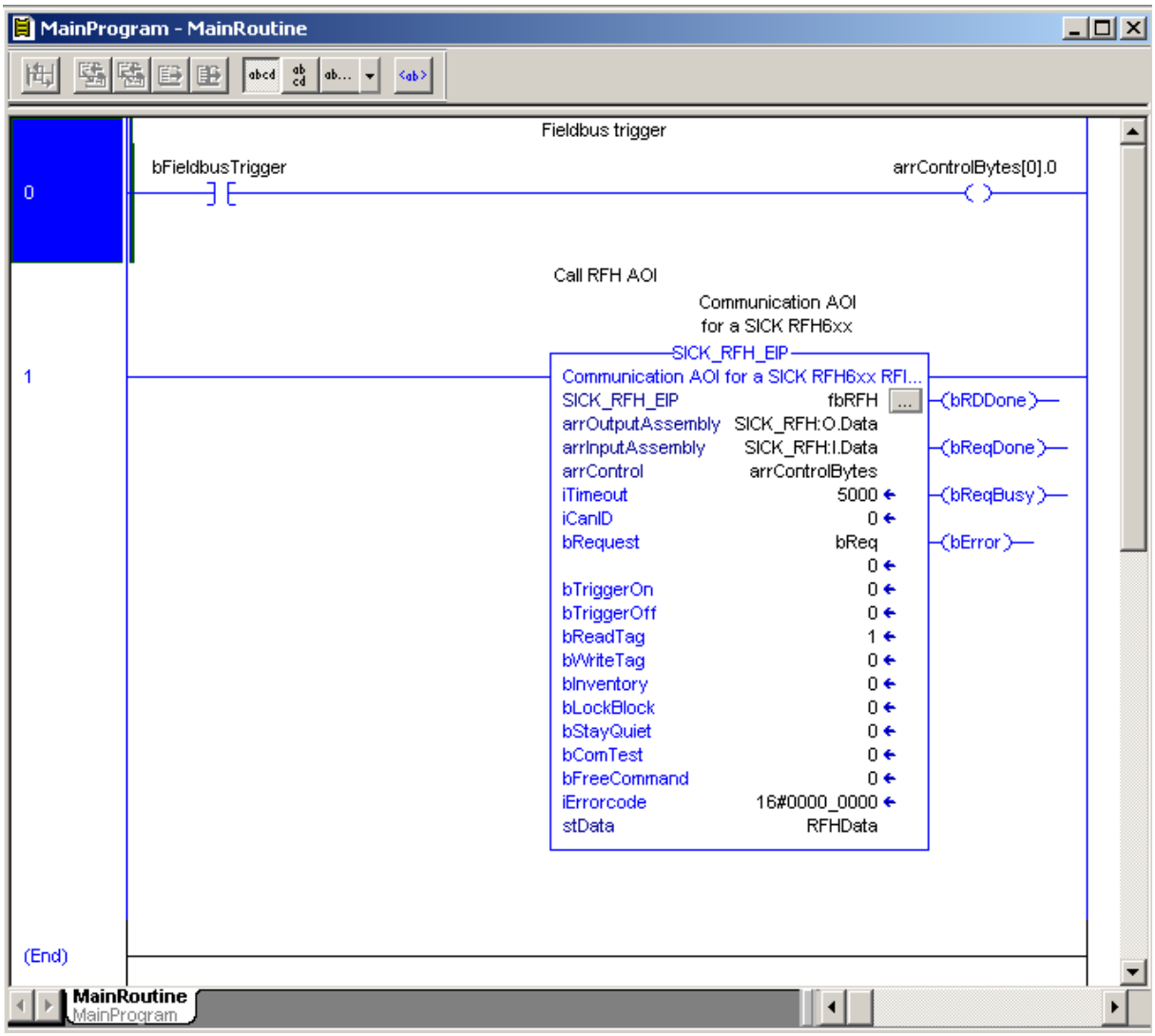


Abbildung 13: Beispielbeschaltung der SICK_RFH_EIP AOI

6.1 Feldbus Trigger

Der RFH kann direkt über das Bit (arrControl[0].0) im Control-Array getriggert werden. Der Baustein empfängt alle Leseergebnisse, unabhängig der gewählten Triggerquelle (Feldbus, Kommando, Sensor1 etc.).

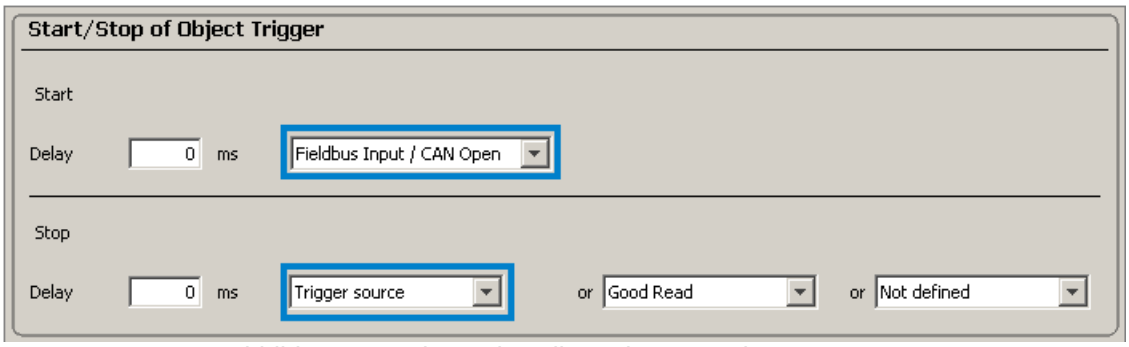


Abbildung 14: Triggereinstellung des RFHs in SOPAS-ET

Der Ausgangsparameter *bRdDone* zeigt für einen SPS Zyklus an, dass neue Daten empfangen wurden. Die vom Gerät gesendeten Daten können im SOPAS Ausgabeformat geändert, bzw. angepasst werden (siehe Kapitel 4.6). Das Triggerergebnis wird in der Variablen *ReadingResult.sResult* der übergebenden Datenstruktur (stData) angezeigt.

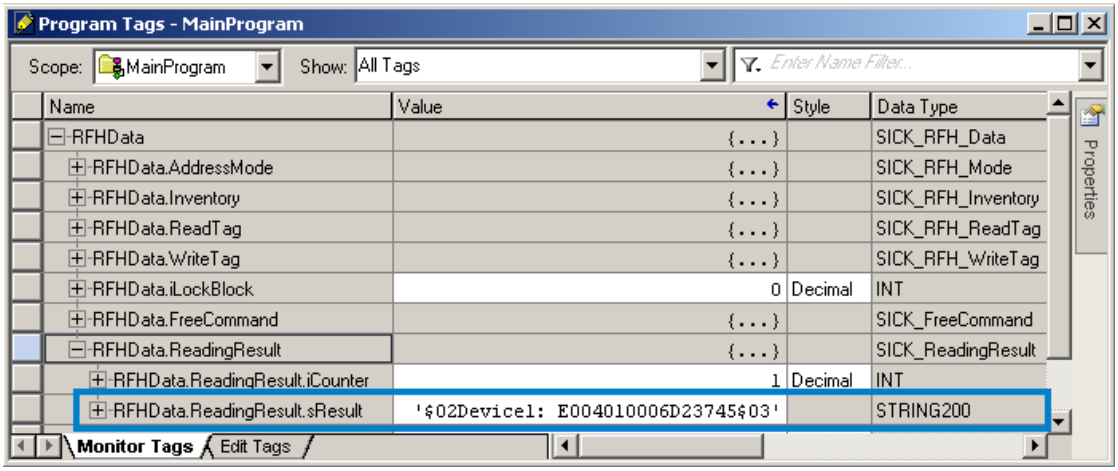


Abbildung 15: Anzeige des Triggerergebnisses

6.2 Auslesen von Tag-Inhalten

Zunächst muss bestimmt werden, mit welchem Transponder kommuniziert werden soll. Ist das Bit *AddressMode.bMode = FALSE*“ wird mit dem Transponder kommuniziert, welcher sich aktuell im Lesebereich des RFHs befindet.

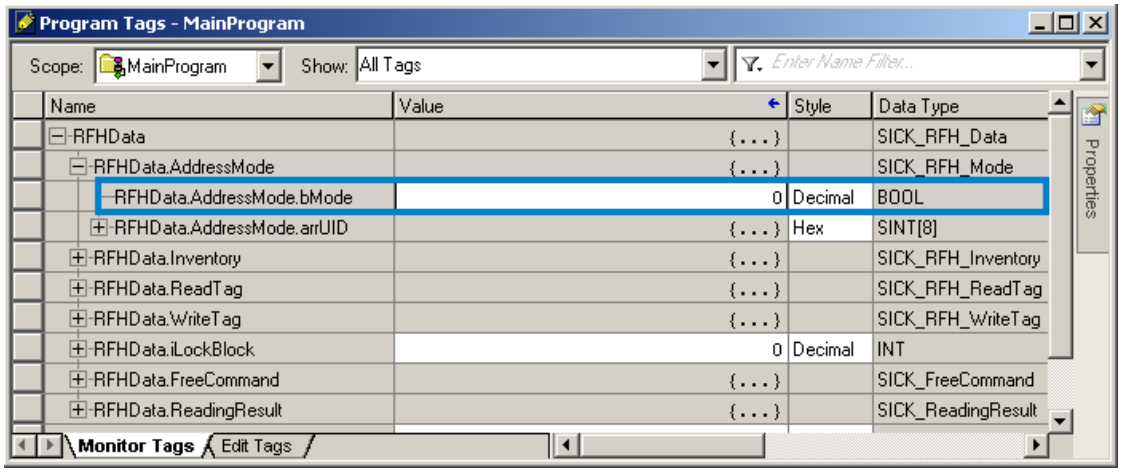


Abbildung 16: Auswahl des Kommunikationsmodus

Anschließend muss definiert werden, welche Inhalte aus dem Transponder ausgelesen werden sollen. In diesem Beispiel sollen 3 Blöcke ausgelesen werden. Da der verwendete Transponder eine Blockgröße von 4 Byte besitzt, werden insgesamt 12 Bytes (3x4 Byte) Daten ab Block 0 ausgelesen.

Start Block:	0	(Erster Block der ausgelesen werden soll)
Anzahl der Blöcke:	3	(Anzahl der Blöcke die gelesen werden sollen)

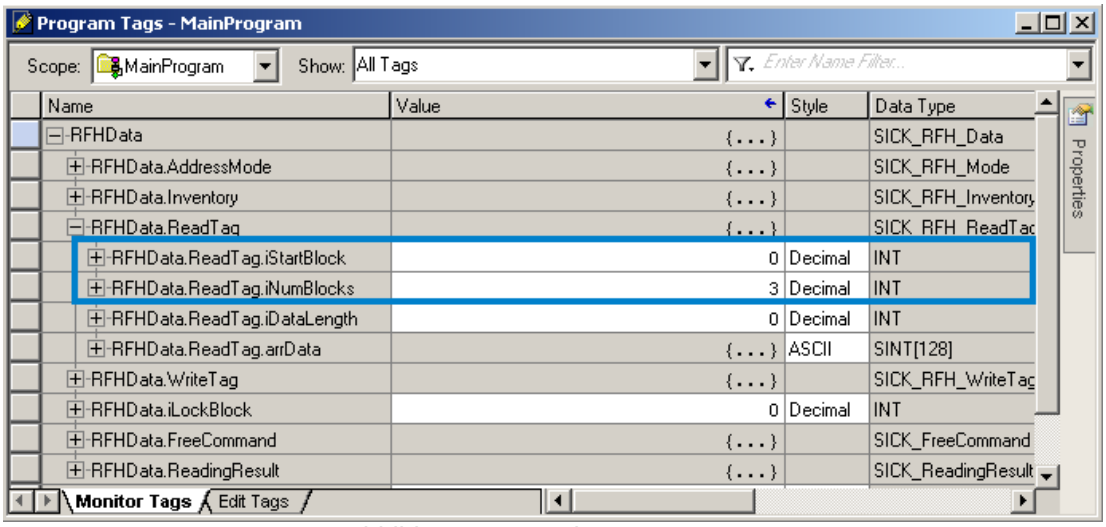


Abbildung 17: Read Tag Parameter

Die Leseaktion (*bReadTag*) wird ausgeführt, sobald das Bit *bRequest* mit einer positiven Flanke angetriggert wird.

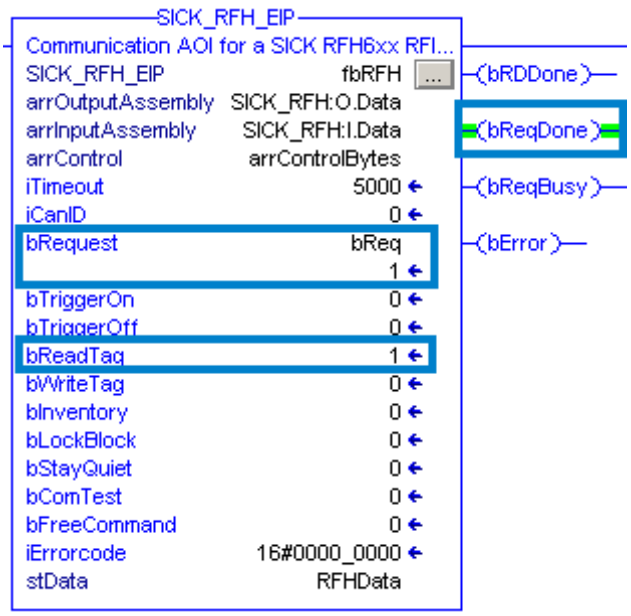


Abbildung 18: Read Tag Aktion starten

Die Leseaktion ist abgeschlossen sobald das Bit *bReqDone* = *TRUE* signalisiert. Der gelesenen Tag-Inhalte stehen im Array *ReadTag.arrData* der übergebenden Datenstruktur zur Verfügung.

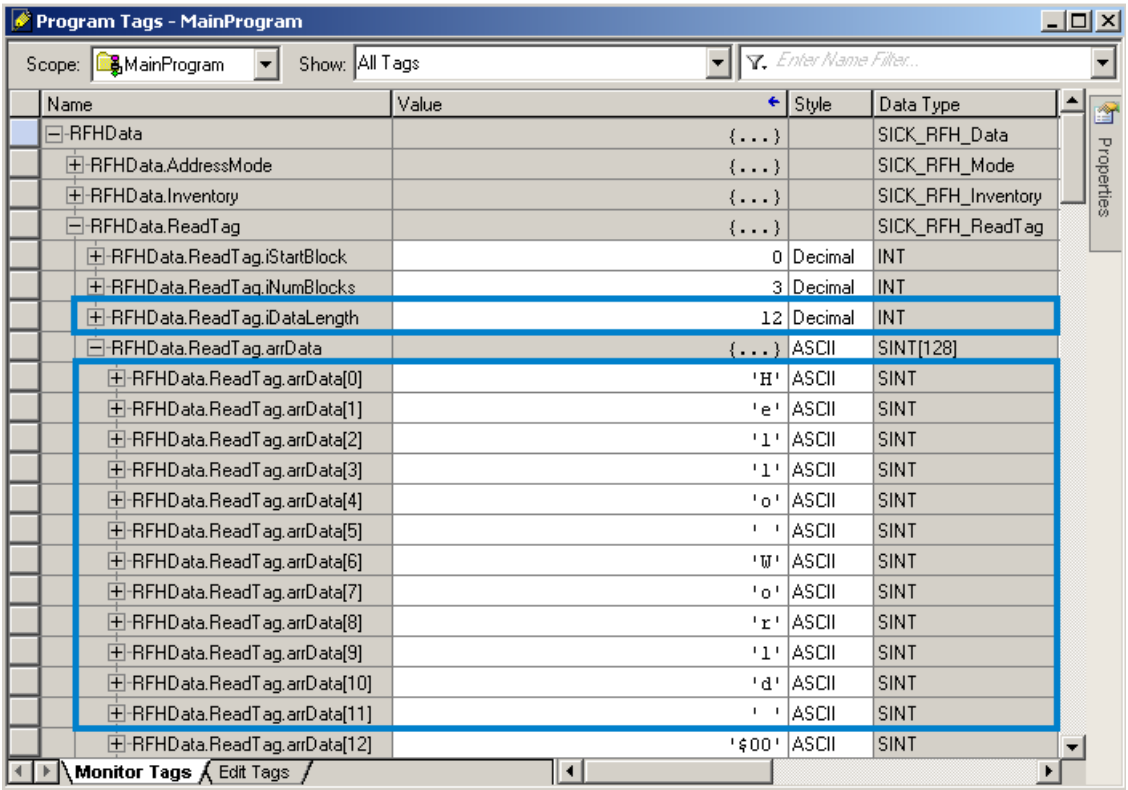


Abbildung 19: Gelesene Tag-Inhalte

6.3 Schreiben von Tag-Inhalten

Zunächst muss bestimmt werden, mit welchem Transponder kommuniziert werden soll. Ist das Bit Address*Mode.bMode = TRUE* kommuniziert der RFH mit dem vorgegebenen Transponder, dessen UID unter dem Parameter *AddressMode.arrUID[]* angegeben wird.

UID: E0 04 01 00 06 D2 37 45 (UID des zu verwendenden Transponders). Die UID muss im Hexadezimal-Format angegeben werden.

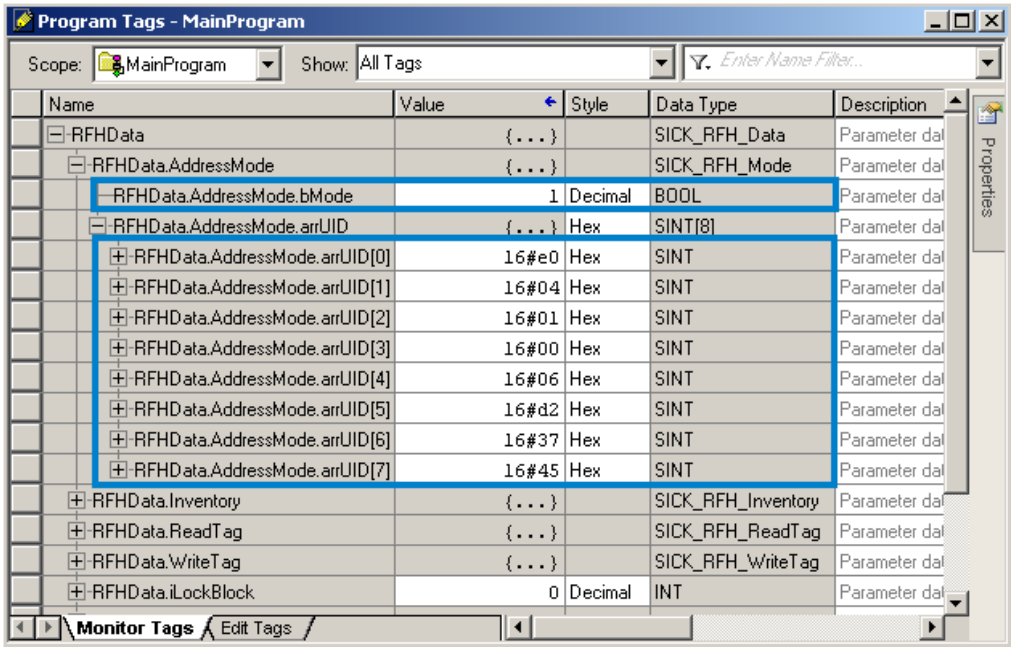


Abbildung 20: Vorgabe der Transponder Identifikation

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

Start Block:	0	(Erster Block der geschrieben werden soll)
Anzahl der Blöcke:	2	(Anzahl der Blöcke die geschrieben werden sollen)
Blockgröße:	4	(Blockgröße des eingesetzten Transponders hier: 4 Byte)
Daten:	’12345678’	Daten die auf den Transponder geschrieben werden sollen. In diesem Beispiel werden 8 Byte Daten auf den Transponder geschrieben.

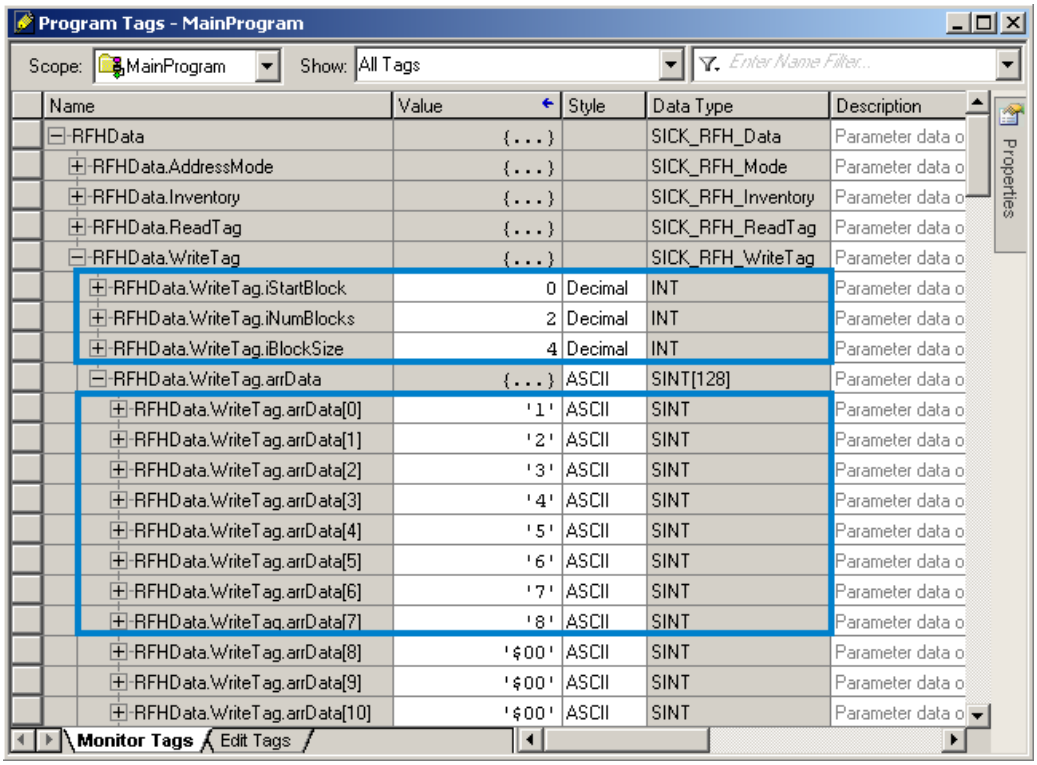


Abbildung 21: Definition der Leseparameter

Die Schreibaktion (*bWriteTag*) wird ausgeführt, sobald das Bit *bRequest* mit einer positiven Flanke angetriggert wird.

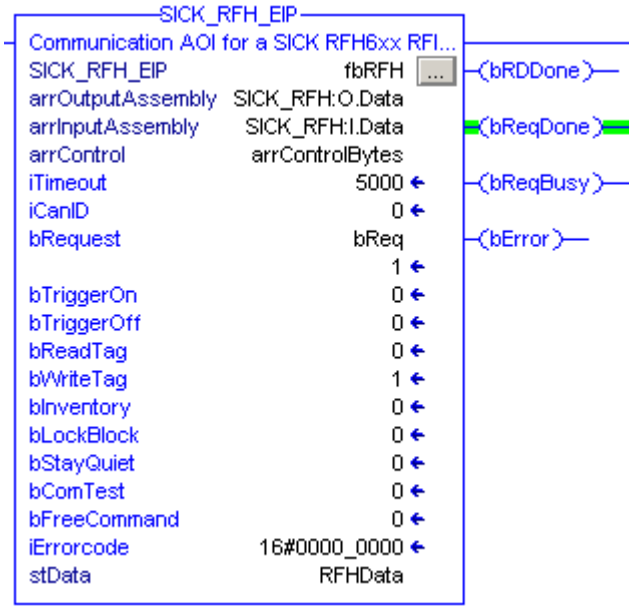


Abbildung 22: Starten der Bausteinfunktion

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