Technische Information

SICK
LECTOR6xx AOI

Bausteinversion V1.X

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

SICK
Sensor Intelligence.

Versionshistorie

Version	Datum	Beschreibung
V1.0	17.12.2012	Initiale Version
V1.1	07.11.2013	Fix arithmetischer Fehler alle 32768 eingehende Leseergebnisse
V1.2	17.08.2015	Fix CCOM Eigenschaften (Visible und Request Einstellungen). Beispielprogram angepasst. Die AOI ist nun unter RSLogix5000 V16 oder höher einsetzbar.
V1.3	27.11.2015	Update CCOM AOI (arithmetischer Fehler alle 128 ausgehenden Telegramme)

Inhaltsverzeichnis

1 Zu diesem Dokument 3
1.1 Funktion dieses Dokuments 3
1.2 Zielgruppe 3
2 Allgemeines 4
3 Einbinden der AOI in RSLogix5000 5
3.1 SOPAS Gerätekonfiguration 5
3.2 Hardwarekonfiguration 6
3.3 AOI Import 9
4 Bausteinbeschreibung 10
4.1 Bausteinspezifikationen 10
4.2 Arbeitsweise 11
4.3 Verhalten im Fehlerfall 11
4.4 Timing 12
4.5 Werteübergabe 13
4.5.1 Matchcode 14
4.5.2 Free Command 15
4.5.3 Reading Result 15
4.6 Triggereinstellungen 16
4.6.1 Trigger über Kommando 16
4.6.2 Feldbus Trigger 17
4.7 Empfangen von Leseergebnissen > 200 Byte 17
5 Parameter 18
6 Fehlercodes 20
7 Beispiel 22
7.1 Feldbus Trigger 23
7.2 Matchcode ändern/anlegen 24

1 Zu diesem Dokument

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

1.1 Funktion dieses Dokuments

Diese Anleitung beschreibt den Umgang mit der SICK_LECTOR_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 Lector 6xx 2D-Code Reader 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_LECTOR_EIP AOI

Bausteinfunktionalitäten:

- Senden eines Triggerbefehls (CoLa[i] Kommando) über die SPS
- Empfang von Leseergebnissen (im SOPAS-ET[ii] Ausgabeformat definiert)
- Anlegen / Ändern einer Evaluationsbedingung für einen Matchcode
- Ausführen eines Kommunikationstests
- Permanentes Speichern aller Geräteparameter im Gerät
- 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_LECTOR_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 Lector zu aktivieren, muss in SOPAS-ET unter dem Menüpunkt *Network / Interfaces /IOs → Ethernet → EtherNet/IP* die folgenden Einstellungen vorgenommen werden:

- EtherNet/IP enabled: Aktivieren
- Communication Mode: with Handshake
- Assembly Output Size at PLC: 10..500
- Assembly Input Size at PLC: 10..500

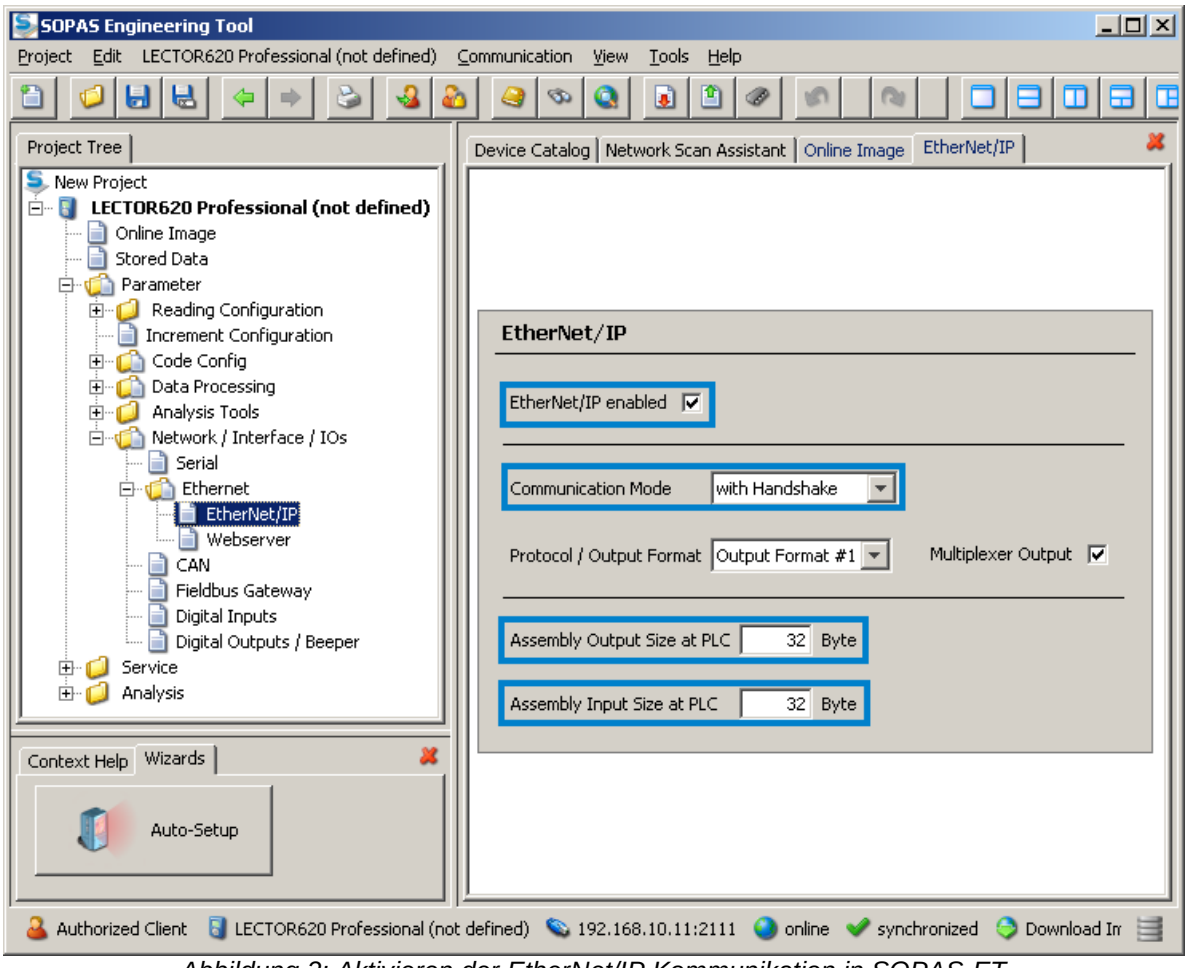


Abbildung 2: Aktivieren der EtherNet/IP Kommunikation in SOPAS-ET

Die AOI kommuniziert über die zyklischen Prozessdaten mit dem Lector (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 Buszyk-

lus übertragen werden können. Die AOI unterstützt Prozessdatenlängen von bis zu 500 Bytes. Ist der Telegraminhalt länger als die Länge der Prozessdaten, wird das Telegram fragmentiert übertragen.

3.2 Hardwarekonfiguration

Um mit RSLogix5000 auf die Input- / Output- Assemblies des Lectors zugreifen zu können, muss dieser zunächst projektiert werden.

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

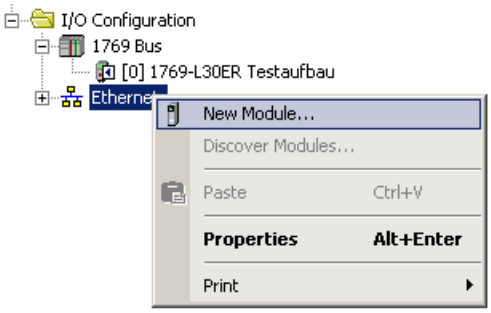


Abbildung 3: Neues Ethernetmodul in RSLogix5000 einfügen

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.

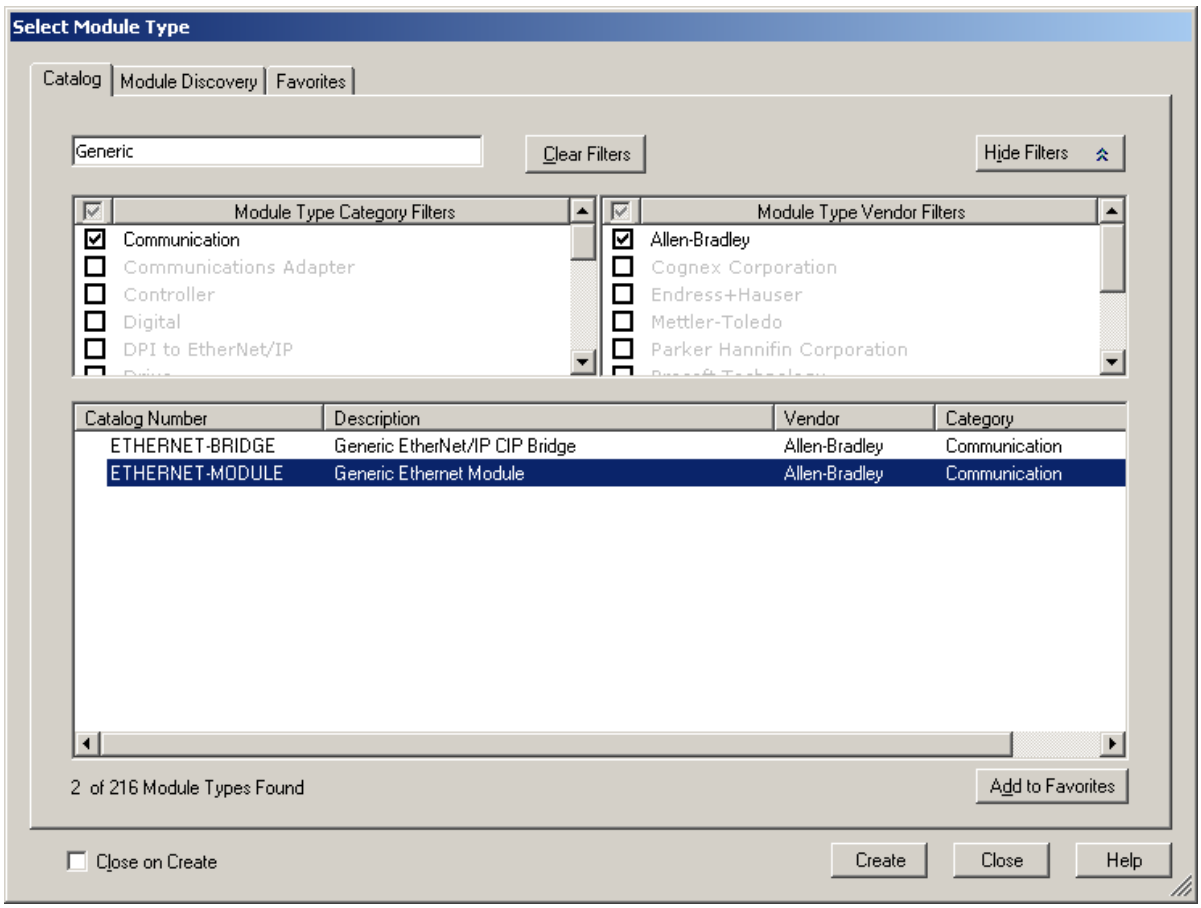


Abbildung 4: Auswahl des Generic Modules in RSLogix5000

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

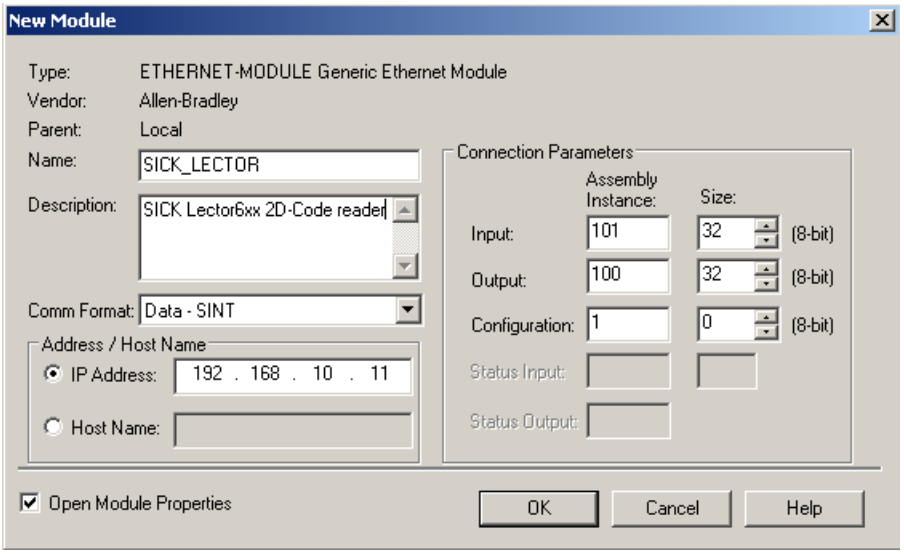


Abbildung 5: Assembly Einstellungen des SICK Lectors

Beispiel:

- Name: SICK_LECTOR (Name ist frei wählbar)
- Comm Format: Data – SINT
- IP Address: 192.168.10.11 (IP-Adresse des SICK Lectors)
- Input Assembly Instance: 101
- Input Assembly Size: 32 (Die Assemblygröße muss mit der in SOPAS konfigurierten Datenlänge identisch sein siehe Kapitel 3.1)
- Output Assembly Instance: 100
- Input Assembly Size: 32 (Die Assemblygröße muss mit der in SOPAS konfigurierten Datenlänge identisch sein siehe Kapitel 3.1)
- Configuration Assembly Instance: 1
- Configuration Assembly Size: 0 (Keine Konfigurations-Assembly vorhanden)

Laden Sie die Konfiguration wie folgt in die Steuerung.

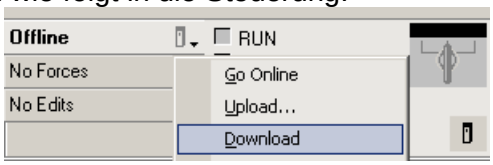


Abbildung 6: Download der SPS Konfiguration

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

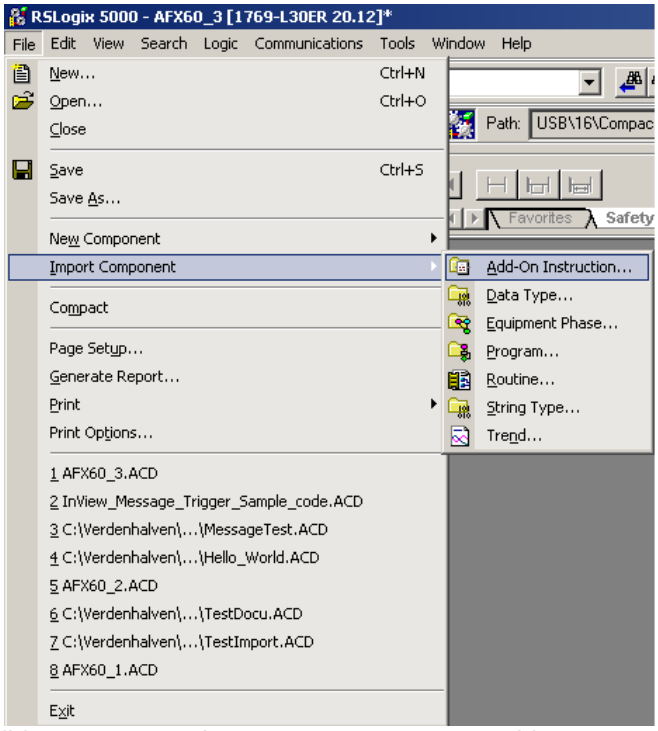


Abbildung 8: Import der SICK_LECTOR_EIP Add-On Instruction

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_EIP“, die die Kommunikation zwischen SPS und Sensor abarbeitet.

4.1 Bausteinspezifikationen

Bausteinname:	SICK_LECTOR_EIP
Version:	1.3
Aufgerufene Bausteine:	SICK_CCOM_EIP
Verwendete UDTs:	SICK_Lector_Data └ SICK_Matchcode └ 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_LECTOR_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_Lector_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
- Match Code → Erstellt / Ändert eine neue Evaluationsbedingung für einen Matchcode
- 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.

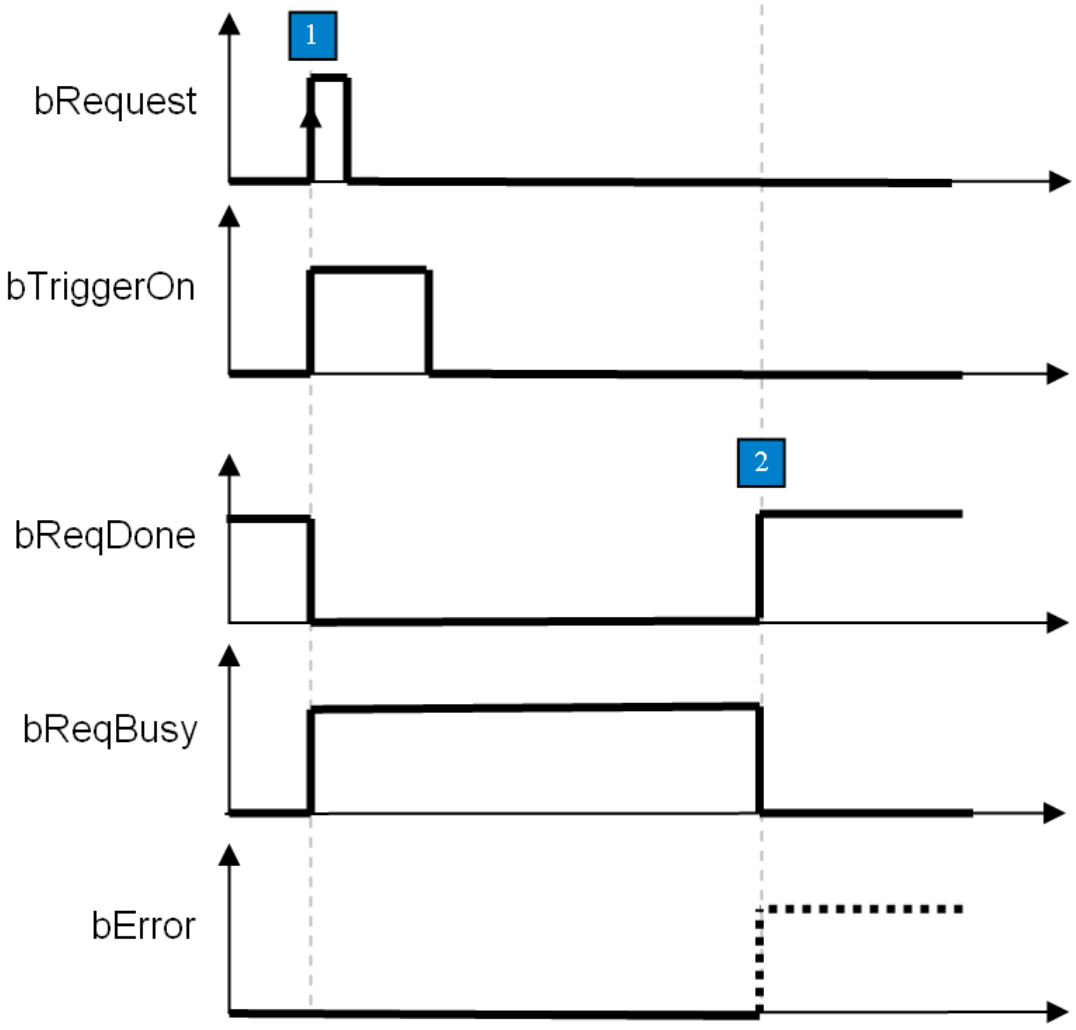
Wenn der Baustein am Ausgangsparameter *bReqDone* = *TRUE* signalisiert, wurde die Aktion erfolgreich durchgeführt. Wurden bei dieser Aktion (z.B. *bFreeCommand*) Daten vom Gerät angefordert, werden diese in den jeweiligen Datenbereich des instanziierten UDTs (*stData*) kopiert.

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_Lector_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.

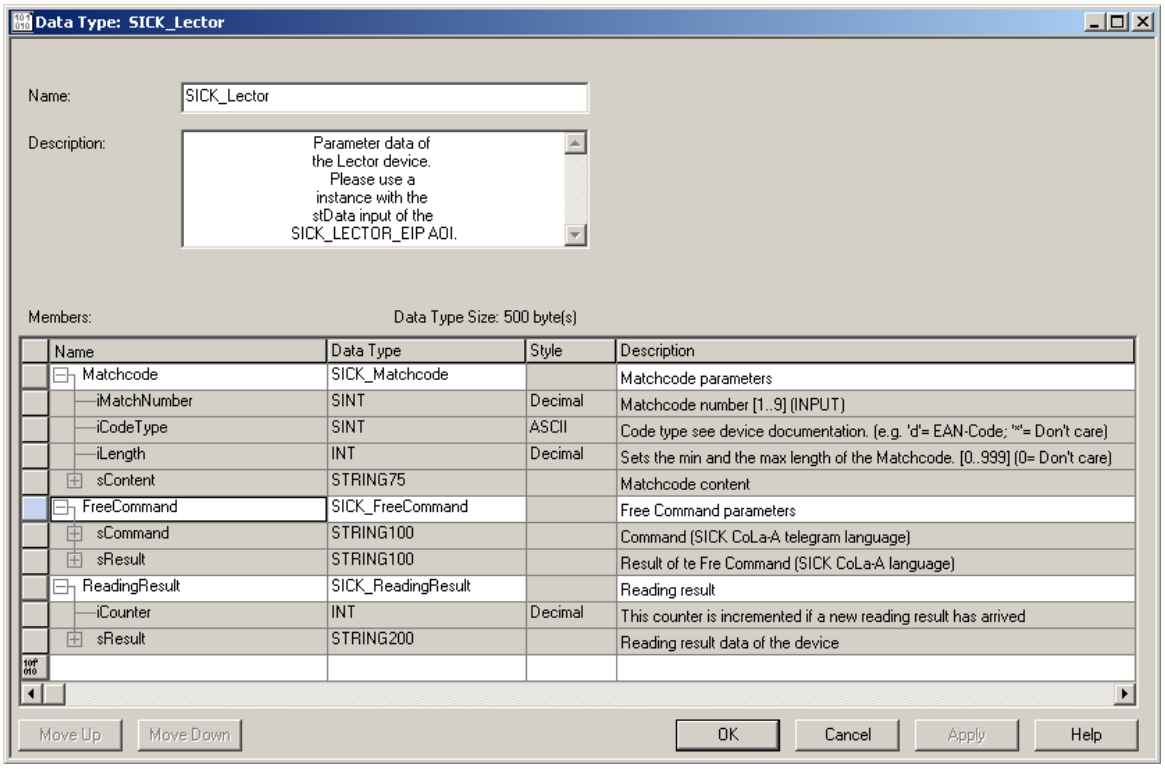


Abbildung 9: Datenstruktur des SICK_Lector_Data UDTs

4.5.1 Matchcode

Mit Hilfe der Matchcode Aktion hat man die Möglichkeit eine Evaluationsbedingung neu zu erstellen, oder eine bereits existierende abzuändern. Bevor die Matchcode Aktion ausgeführt werden kann, müssen in der Struktur *Matchcode* die folgenden Parameter angegeben werden.

Parameter	Deklaration	Datentyp	Beschreibung
Matchcode. iMatchNumber	Input	SINT	Mit der Matchcode Nummer wird der Name der Evaluationsbedingung festgelegt (z.B. 1=Match1, 2=Match2). Gültiger Wertbereich: [1..9]
Matchcode. iCodeType	Input	SINT (ASCII)	Gewünschter Code Typ, auf der sich die Evaluationsbedingung beziehen soll (z.B. ’w’= Datamatrix, ’s’= QR-Code, ’*’ = Beliebiger Code Typ). Eine Auswahl aller Codetypen siehe Gerätedokumentation. Gültiger Wertbereich: [33..126]
Matchcode. iLength	Input	INT	Minimale und maximale Codelänge. 0 = Beliebige Codelänge Gültiger Wertebereich: [0..999]
Matchcode. sContent	Input	STRING 75	Matchcodeinhalt

Tabelle 1: Matchcode Parameter

4.5.2 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 2: Free Command Parameter

4.5.3 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 0 beschreibt dass Vorgehen beim Empfang von längeren Datentelegrammen.

Tabelle 3: Reading Result Parameter

4.6 Triggereinstellungen

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

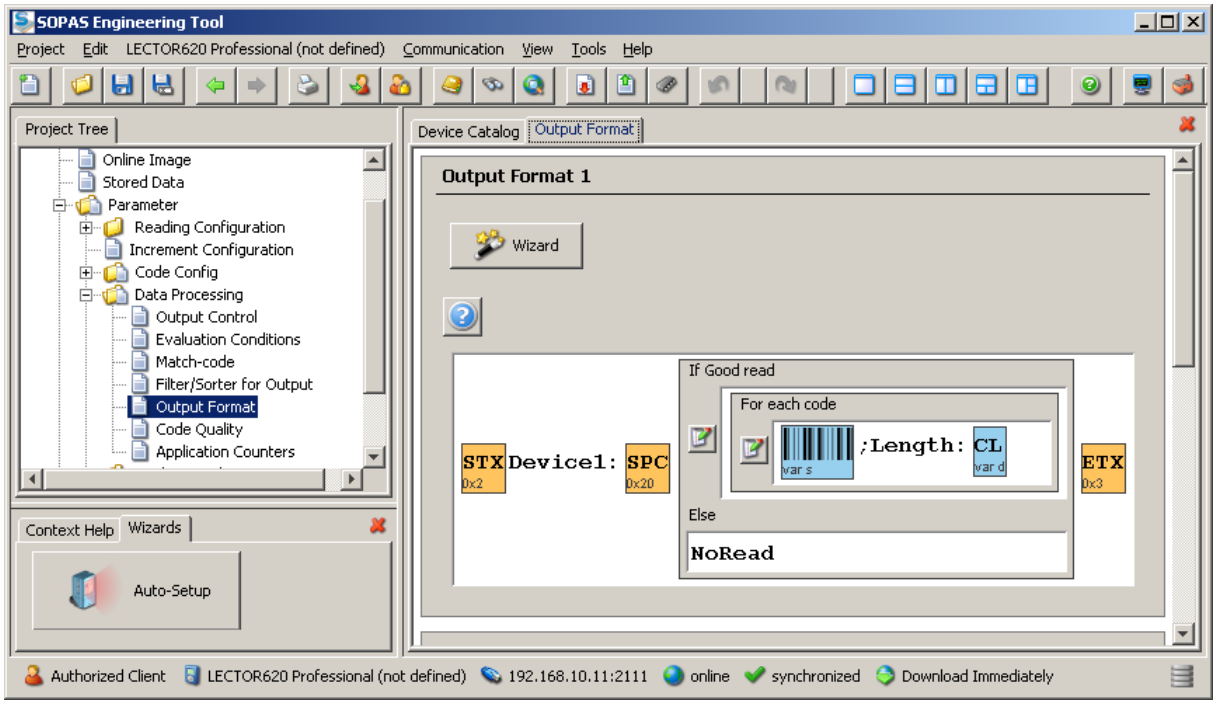


Abbildung 10: Beispielkonfiguration des Ausgabeformats in SOPAS-ET

Der Lector kann auf verschiedene Arten getriggert werden.

- Softwaretrigger über die AOI (*bTriggerOn* / *bTriggerOff*)
- Feldbustrigger über die AOI (*arrControl*)
- Hardwaretrigger über den Sensor1 Eingang des Lectors
- 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 „Objekt Trigger“ der Lector 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).

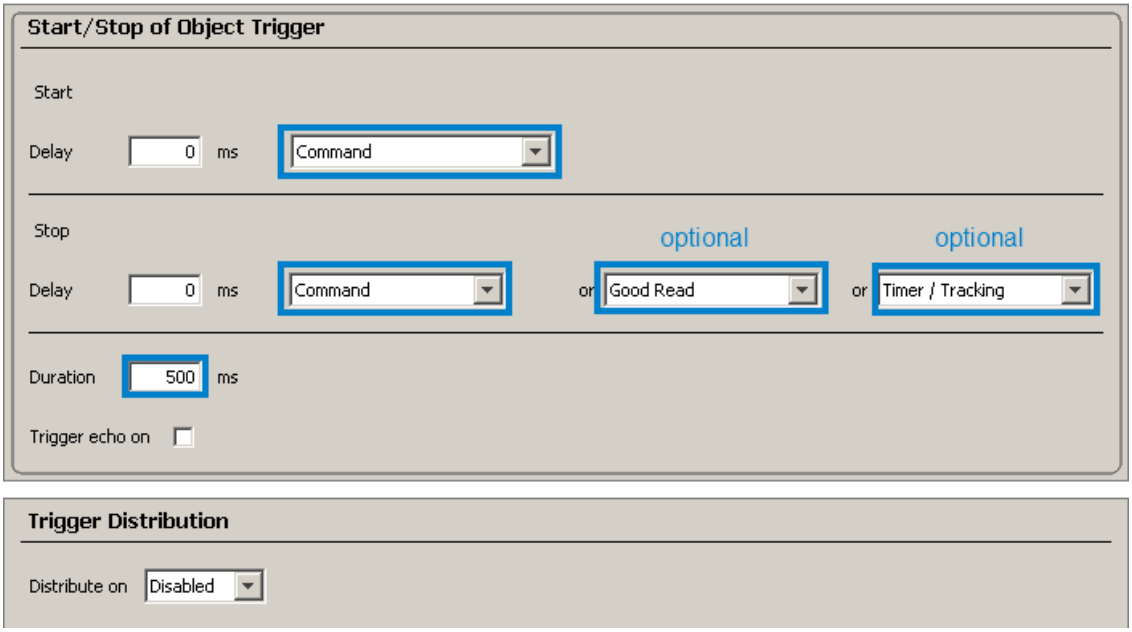


Abbildung 11: SOPAS Trigger Einstellungen

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 Lector 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_Lector_Data UDT:</u>
In der mitgelieferten UDT (SICK_Lector_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.

5 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. Beispiel: arrInputAssembly:= myLector: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. Beispiel: arrOutputAssembly:= myLector: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. Vorraussetzung 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_Lector_Data) abgelegt.

Parameter	Deklaration	Datentyp	Beschreibung
bMatchcode	INPUT	BOOL	Bausteinaktion: Erstellen einer Matchcode Bedingung. Die Aktion setzt voraus, dass in der Struktur (Matchcode) die im Kapitel 4.5.1 beschriebenen Parameter angegebenen werden.
bComTest	INPUT	BOOL	Bausteinaktion: Ausführen eines Kommunikationstests. bReqDone= TRUE: Kommunikation OK bReqDone= FALSE: Kommunikation nicht OK
bSave Permanent	INPUT	BOOL	Bausteinaktion: Permanentes Speichern aller Geräteparameter im Gerät.
bFree Command	INPUT	BOOL	Bausteinaktion: Ausführen eines freien Kommandos. Die Aktion setzt ein gültiges CoLa-Kommando in der Datenstruktur (FreeCommand. sCommand) vorraus (siehe Kapitel 4.5.2). Die Kommandoantwort steht nach einer erfolgreichen Übertragung (bReqDone=TRUE) im Result-String der Datenstruktur zur Verfügung.
stData	IN/OUT	SICK_ Lector_ Data	Übergabe der zugehörigen UDT Struktur (SICK_Lector_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 4: Bausteinparameter

6 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_EIP Bausteins (siehe Bausteindokumentation).
16#00**XX**_000A	Gerätefehler	Es ist ein Gerätefehler aufgetreten (’sFA XX’) **XX** = Gerätefehler (siehe Gerätedokumentation)
16#0000_000B	Ungültige Kommandoantwort	Die gewählte Aktion wurde nicht ausgeführt. Dies kann je nach Aktion die folgenden Ursachen haben: - Triggereinstellung in der SOPAS Objekt Trigger Einstellung fehlerhaft - Gerät befindet sich nicht im „Run-Mode“
16#0000_000C … 16#0000_000F	Reserviert	Reserviert

Fehlercode	Kurzbeschreibung	Beschreibung
16#0000_0010	9 < Match-code.iMatchNumber < 1	Ungültige Matchode Nummer. Gültiger Wertebereich: [1..9]
16#0000_0011	999 < Matchcode. iLength < 0	Ungültige Min/Max Länge des Matchcode. Gültiger Wertebereich: [0..999]
16#**XXXX**_0012	Matchcode / Save Per-manent wurde nicht aus-geführt.	Die gewählte Aktion wurde nicht ausgeführt. Das Gerät wurde wieder in den „RUN-Mode“ versetzt. **XXXX** = Fehlercode des aufgetretenen Feh-lers
16#0000_0013	Wechsel in den „RUN-Mode“ nicht möglich.	Das Gerät kann nicht zurück in den „RUN-Mode“ versetzt werden. Dies könnte folgende Ursache haben: - Kommandoverarbeitung dauert länger als die eingestellte Timeout-Zeit der Routine.
16#0000_0014	126 < iCodeType < 33	Ungültiger Code Type. Bitte entnehmen Sie die Werte für die Code Typen der Gerätebe-schreibung. Gültiger Wertebereich: [33..126]
16#0000_0015	75 < Matchcode content < 1	Ungültige länge des Matchcode Inhalts. Gültiger Wertebereich: [1..75]
sReadResult.LEN = -1	Eingehendes Leseergeb-nis > 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 emp-fangen von Leserergebnissen größer 200 Byte siehe Kapitel 4.7.

Tabelle 5: Fehlercodes

7 Beispiel

Abbildung 13 zeigt eine Beispielbeschaltung der SICK_LECTOR_EIP AOI. Da sich das Gerät nicht in einem 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:

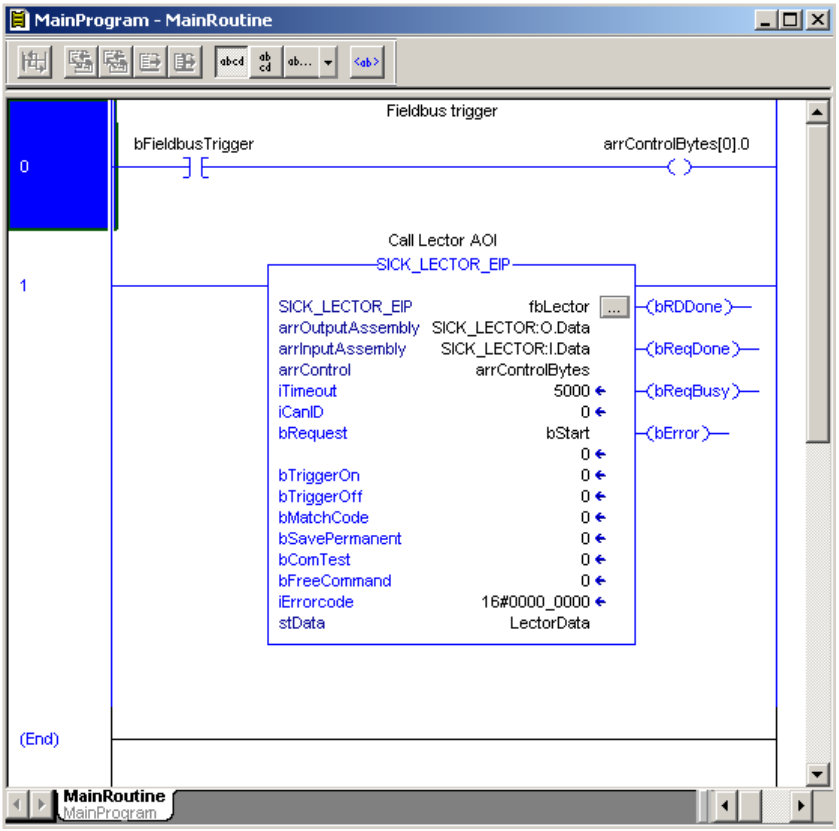


Abbildung 13: Beispielbeschaltung der SICK_LECTOR_EIP AOI

7.1 Feldbus Trigger

Der Lector 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.).

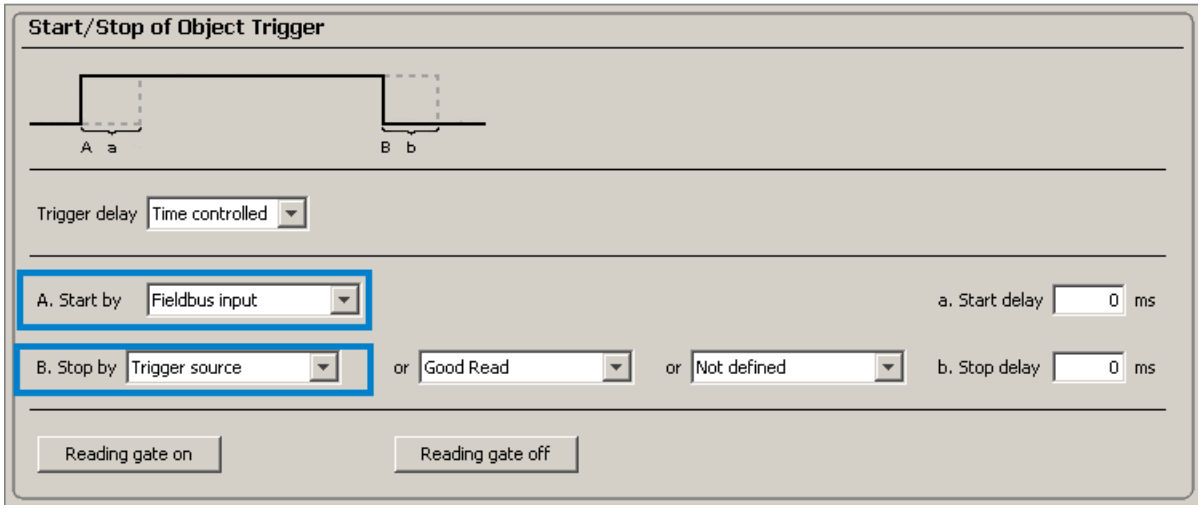


Abbildung 14: Triggereinstellung des Lectors in SOPAS-ET

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.

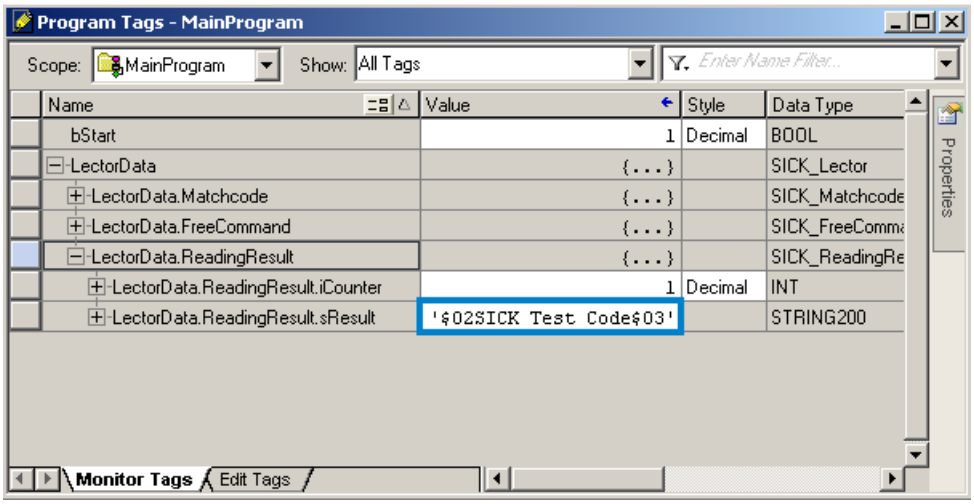


Abbildung 15: Anzeige des Triggerergebnisses

7.2 Matchcode ändern/anlegen

Um eine neue Matchcode Evaluationsbedingung anzulegen bzw. eine existierende zu ändern müssen zunächst die erforderlichen Parameterwerte in der übergebenden Datenstruktur (stData) angegeben werden.

iMatchNumber:	Matchcode Nummer (hier: Match5)
iCodeType:	Code Typ (“*“= Alle Code Typen)
iLength	Min / Max Länge des Matchcodes (hier: 20 Zeichen)
sContent:	Codeinhalt (hier: ‚12345*’)

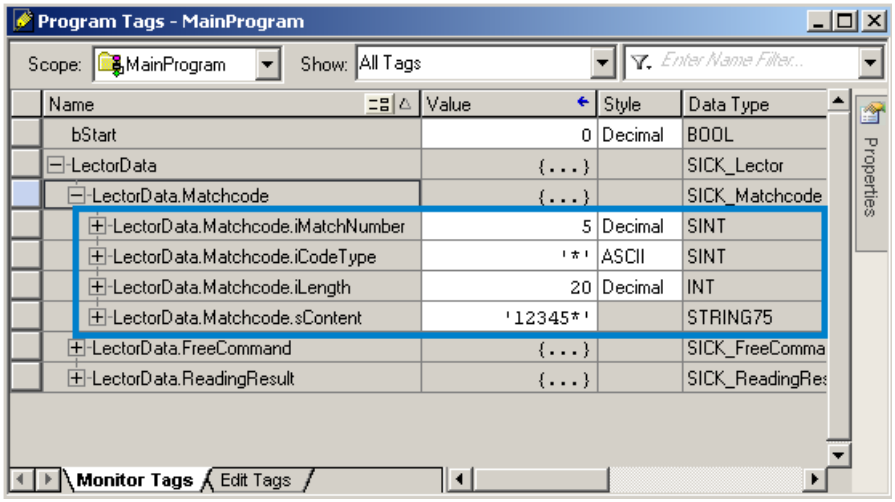


Abbildung 16: Machcode Parameter

Die Matchcode Aktion *bMatchcode* wird ausgeführt, sobald das Bit *bRequest* mit einer positiven Flanke angetriggert wird.

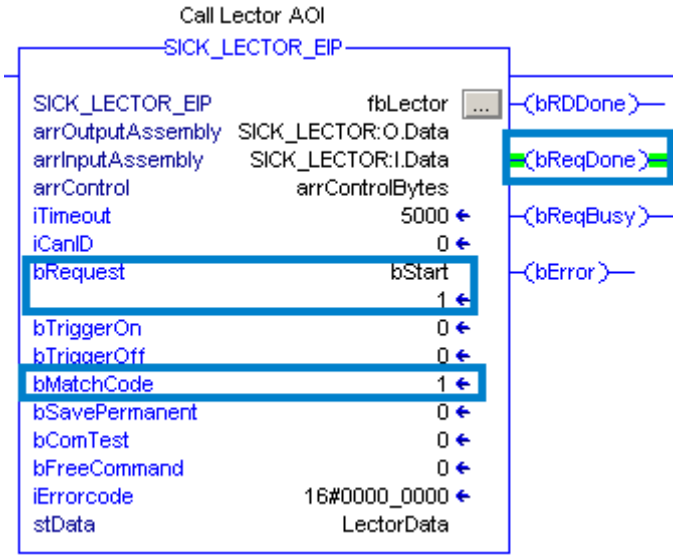


Abbildung 17: Matchcode Aktion starten

Die Matchcodeaktion ist abgeschlossen sobald das Bit *bReqDone* = *TRUE* signalisiert. In diesem Beispiel wird auf das Gerät die folgende Evaluationsbedingung angelegt.

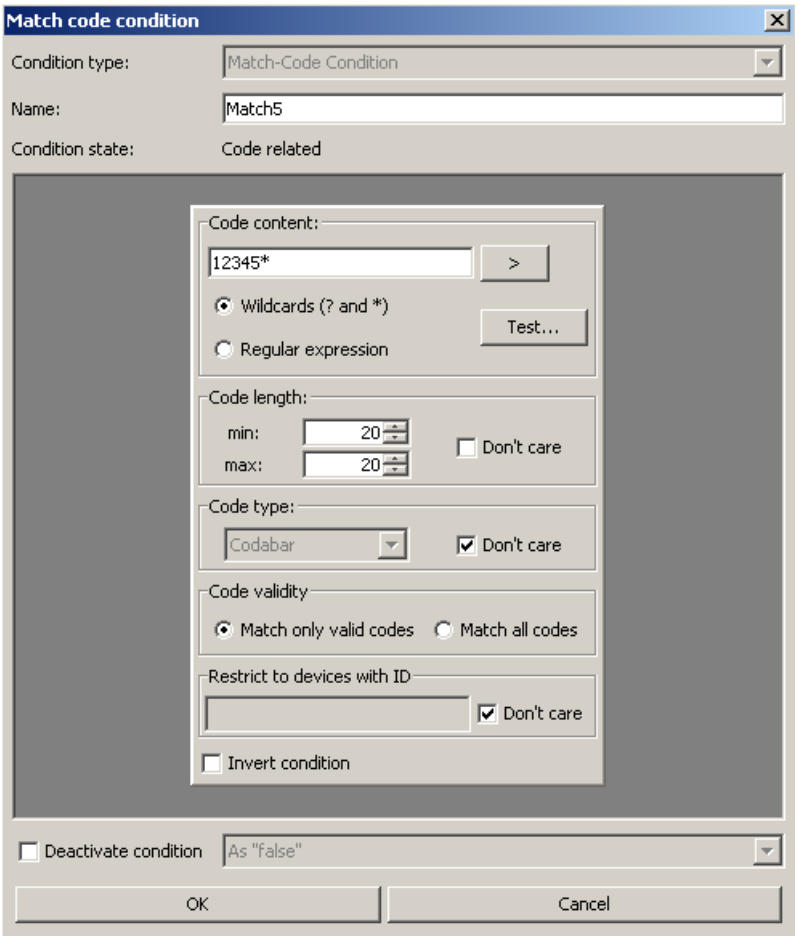


Abbildung 18: Angelegte Evaluationsbedingung