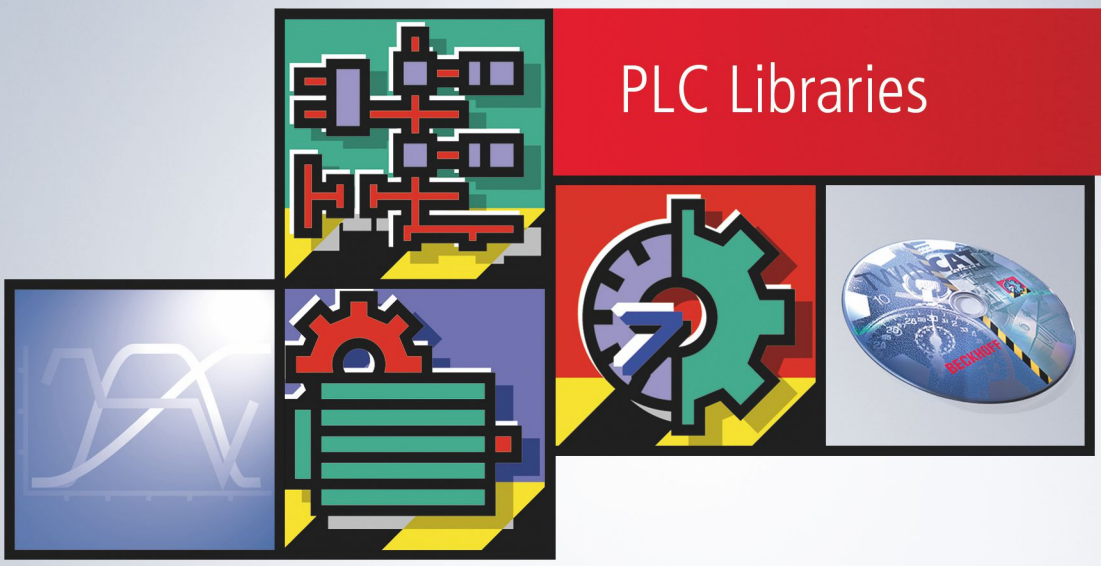
BECKHOFF New Automation Technology

Handbuch | DE

TX1200

TwinCAT 2 | PLC-Bibliothek: TcAdsBC



10.11.2022 | Version: 1.1

Inhaltsverzeichnis

1 Vorwort

1.1 Hinweise zur Dokumentation

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs- und Automatisierungstechnik, das mit den geltenden nationalen Normen vertraut ist.
Zur Installation und Inbetriebnahme der Komponenten ist die Beachtung der Dokumentation und der nachfolgenden Hinweise und Erklärungen unbedingt notwendig.
Das Fachpersonal ist verpflichtet, für jede Installation und Inbetriebnahme die zu dem betreffenden Zeitpunkt veröffentliche Dokumentation zu verwenden.

Das Fachpersonal hat sicherzustellen, dass die Anwendung bzw. der Einsatz der beschriebenen Produkte alle Sicherheitsanforderungen, einschließlich sämtlicher anwendbaren Gesetze, Vorschriften, Bestimmungen und Normen erfüllt.

Disclaimer

Diese Dokumentation wurde sorgfältig erstellt. Die beschriebenen Produkte werden jedoch ständig weiter entwickelt.
Wir behalten uns das Recht vor, die Dokumentation jederzeit und ohne Ankündigung zu überarbeiten und zu ändern.
Aus den Angaben, Abbildungen und Beschreibungen in dieser Dokumentation können keine Ansprüche auf Änderung bereits gelieferter Produkte geltend gemacht werden.

Marken

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® und XPlanar® sind eingetragene und lizenzierte Marken der Beckhoff Automation GmbH.
Die Verwendung anderer in dieser Dokumentation enthaltenen Marken oder Kennzeichen durch Dritte kann zu einer Verletzung von Rechten der Inhaber der entsprechenden Bezeichnungen führen.

Patente

Die EtherCAT-Technologie ist patentrechtlich geschützt, insbesondere durch folgende Anmeldungen und Patente:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
mit den entsprechenden Anmeldungen und Eintragungen in verschiedenen anderen Ländern.

EtherCAT®

EtherCAT® ist eine eingetragene Marke und patentierte Technologie lizenziert durch die Beckhoff Automation GmbH, Deutschland

Copyright

1.2 Sicherheitshinweise

Sicherheitsbestimmungen

Beachten Sie die folgenden Sicherheitshinweise und Erklärungen!
Produktspezifische Sicherheitshinweise finden Sie auf den folgenden Seiten oder in den Bereichen Montage, Verdrahtung, Inbetriebnahme usw.

Haftungsausschluss

Die gesamten Komponenten werden je nach Anwendungsbestimmungen in bestimmten Hard- und Software-Konfigurationen ausgeliefert. Änderungen der Hard- oder Software-Konfiguration, die über die dokumentierten Möglichkeiten hinausgehen, sind unzulässig und bewirken den Haftungsausschluss der Beckhoff Automation GmbH & Co. KG.

Qualifikation des Personals

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs-, Automatisierungs- und Antriebstechnik, das mit den geltenden Normen vertraut ist.

Erklärung der Symbole

In der vorliegenden Dokumentation werden die folgenden Symbole mit einem nebenstehenden Sicherheitshinweis oder Hinweistext verwendet. Die Sicherheitshinweise sind aufmerksam zu lesen und unbedingt zu befolgen!

⚠ GEFAHR

Akute Verletzungsgefahr!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, besteht unmittelbare Gefahr für Leben und Gesundheit von Personen!

⚠ WARNUNG

Verletzungsgefahr!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, besteht Gefahr für Leben und Gesundheit von Personen!

⚠ VORSICHT

Schädigung von Personen!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, können Personen geschädigt werden!

HINWEIS

Schädigung von Umwelt oder Geräten

Wenn der Hinweis neben diesem Symbol nicht beachtet wird, können Umwelt oder Geräte geschädigt werden.

Tipp oder Fingerzeig

Dieses Symbol kennzeichnet Informationen, die zum besseren Verständnis beitragen.

1.3 Hinweise zur Informationssicherheit

Die Produkte der Beckhoff Automation GmbH & Co. KG (Beckhoff) sind, sofern sie online zu erreichen sind, mit Security-Funktionen ausgestattet, die den sicheren Betrieb von Anlagen, Systemen, Maschinen und Netzwerken unterstützen. Trotz der Security-Funktionen sind die Erstellung, Implementierung und ständige Aktualisierung eines ganzheitlichen Security-Konzepts für den Betrieb notwendig, um die jeweilige Anlage, das System, die Maschine und die Netzwerke gegen Cyber-Bedrohungen zu schützen. Die von Beckhoff verkauften Produkte bilden dabei nur einen Teil des gesamtheitlichen Security-Konzepts. Der Kunde ist dafür verantwortlich, dass unbefugte Zugriffe durch Dritte auf seine Anlagen, Systeme, Maschinen und Netzwerke verhindert werden. Letztere sollten nur mit dem Unternehmensnetzwerk oder dem Internet verbunden werden, wenn entsprechende Schutzmaßnahmen eingerichtet wurden.

Zusätzlich sollten die Empfehlungen von Beckhoff zu entsprechenden Schutzmaßnahmen beachtet werden. Weiterführende Informationen über Informationssicherheit und Industrial Security finden Sie in unserem https://www.beckhoff.de/secguide.

Die Produkte und Lösungen von Beckhoff werden ständig weiterentwickelt. Dies betrifft auch die Security-Funktionen. Aufgrund der stetigen Weiterentwicklung empfiehlt Beckhoff ausdrücklich, die Produkte ständig auf dem aktuellen Stand zu halten und nach Bereitstellung von Updates diese auf die Produkte aufzuspielen. Die Verwendung veralteter oder nicht mehr unterstützter Produktversionen kann das Risiko von Cyber-Bedrohungen erhöhen.

Um stets über Hinweise zur Informationssicherheit zu Produkten von Beckhoff informiert zu sein, abonnieren Sie den RSS Feed unter https://www.beckhoff.de/secinfo.

2 Übersicht

Die Bibliothek beinhaltet Funktionsbausteine für eine azyklische Client-Server-Kommunikation zwischen einem Bus-Controller **BC9xxx** und anderen ADS-Geräten im Netzwerk. Die Funktionsweise der Bausteine unterscheidet sich im Wesentlichen nicht von denen für das SPS-Laufzeitsystem auf dem PC.

Voraussetzungen

Die oberen 4 Zahlen der Ams-NetID (Netzwerkadresse) und die TCP/IP Adresse der ADS-Zielgeräte müssen übereinstimmen um die IP-Verbindung aufzubauen.

Beispiel:

Es soll eine ADS-Kommunikation mit einer Profibus-Karte FC310x aufgebaut werden. Der Karte wird bei der Konfiguration vom TwinCAT System Manager eine eigene Netzwerkadresse zugewiesen z.B.: **'172.16.2.209.4.1'**. Damit kann sie als ein eigenständiges ADS-Gerät (Remote PC) betrachtet werden. Damit die IP-Verbindung aufgebaut werden kann, müssen die ersten 4 Stellen der Netzwerkadresse mit der TCP/IP-Adresse des PC's übereinstimmen in dem diese Karte konfiguriert wurde. D.h. die TCP/IP-Adresse des PC's muss: **'172.16.2.209'** lauten.

Bemerkungen

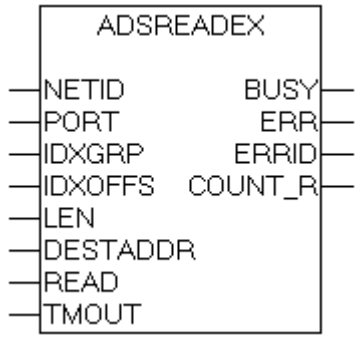
- Um den Ressourcenverbrauch zu minimieren, wurde die maximale Anzahl der gleichzeitig geöffneten Verbindungen auf 4 begrenzt.
- Eine Verbindung wird beim Absenden von einem ADS-Kommando automatisch aufgebaut.
- Eine Verbindung wird nach ca. 10 Sekunden automatisch getrennt, wenn sie in dieser Zeit nicht benutzt wurde.Eine nicht benötigte Verbindung kann mit dem ADSCLOSE-Funktionsbaustein vor dem Ablauf dieser Zeit explizit getrennt werden.
- Die maximale Datengrösse, die ein Buscontroller BC9xxx senden (z.B. mit ADSWRITE oder ADSREADRESBC) und empfangen kann (z.B. mit ADSREAD oder ADSWRITEIND) ist auf 1980 Byte begrenzt.

ADS-Funktionsbausteine

Name	Beschreibung
ADSREADEX [▶ 9]	Daten von einem ADS-Gerät lesen
ADSRDWRTEX [▶ 11]	Daten zu einem ADS-Gerät schreiben und lesen
ADSWRITE [▶ 13]	Daten zu einem ADS-Gerät schreiben
ADSCLOSE [▶ 15]	Die IP-Verbindung zu einem anderen ADS-Gerät explizit schließen
ADSREADIND [▶ 17]	ADSREAD-Indication.
ADSWRITEIND [▶ 18]	ADSWRITE-Indication.
ADSRDWRTIND [▶ 20]	ADSRDWRT-Indication.
ADSREADRESBC [▶ 21]	ADSREAD-Response.
ADSWRITERESBC [▶ 22]	ADSWRITE-Response.
ADSRDWRTRESBC [▶ 23]	ADSRDWRT-Response.

3 Funktionsbausteine

3.1 ADSREADEX



Der Funktionsbaustein erlaubt die Ausführung eines ADS-Lesebefehls, um Daten von einem ADS-Gerät anzufordern.

VAR_INPUT

```
VAR_INPUT
    NETID       : T_AmsNetId;
    PORT        : T_AmsPort;
    IDXGRP      : UDINT;
    IDXOFFS     : UDINT;
    LEN         : UDINT;
    DESTADDR    : DWORD;
    READ        : BOOL;
    TMOUT       : TIME;
END_VAR
```

T_AmsNetId [▶ 31]

T_AmsPort [▶ 31]

NETID : Ist ein String, der die AMS-Netzwerkkennung des Zielgerätes enthält, an das der ADS-Befehl gerichtet wird.

PORT : Enthält die Portnummer des ADS-Gerätes.

IDXGRP : Enthält die Index-Gruppennummer (32bit, unsigned) des angeforderten ADS-Dienstes. Dieser Wert ist der ADS-Tabelle des angesprochenen Gerätes zu entnehmen.

IDXOFFS : Enthält die Index-Offsetnummer (32bit, unsigned) des angeforderten ADS-Dienstes. Dieser Wert ist der ADS-Tabelle des angesprochenen Gerätes zu entnehmen.

LEN : Enthält die Anzahl der zu lesenden Daten in Bytes.

DESTADDR : Enthält die Adresse des Puffers, der die gelesenen Daten aufnehmen soll. Der Programmierer ist selbst dafür verantwortlich den Puffer in der Größe so zu dimensionieren, dass er ‚LEN' Bytes aufnehmen kann. Der Puffer kann eine Einzelvariable, ein Array oder eine Struktur sein, dessen Adresse man mit dem ADR - Operator ermitteln kann.

READ : Durch eine steigende Flanke an diesem Eingang wird der ADS-Befehl ausgelöst.

TMOUT : Gibt die Zeit bis zum Abbrechen der Funktion an.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY        : BOOL;
    ERR         : BOOL;
```

```
    ERRID       : UDINT;
    COUNT_R     : UDINT;
END_VAR
```

BUSY : Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

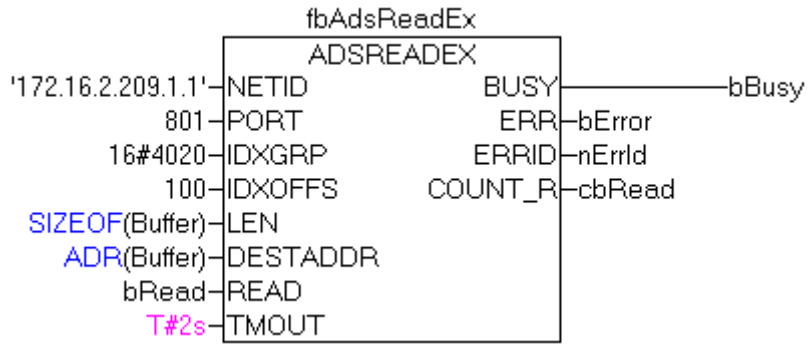
ERR : Sollte ein ADS-Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

ERRID : Liefert bei einem gesetzten ERR-Ausgang die ADS- oder die Gerätespezifische-Fehlernummer [▶ 33].

COUNT_R: Anzahl der erfolgreich gelesenen Datenbytes.

Beispiel für einen Aufruf in FUP:

```
PROGRAM MAIN
VAR
    fbAdsReadEx : ADSREADEX;
    bRead       : BOOL;
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
    cbRead      : UDINT;
    Buffer      : ARRAY[1..10] OF BYTE;
END_VAR
```

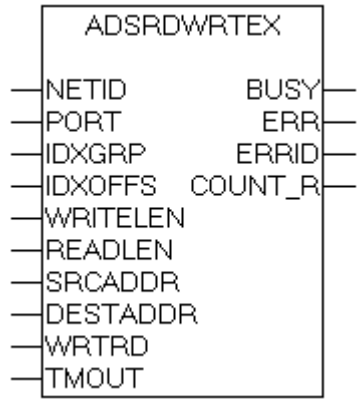


Im Beispiel wird ein ADS-Lesekommando an ein ADS-Gerät mit der Netzwerkadresse: '172.16.2.209.1.1' und der Portnummer 801 gesendet. Über diese Portnummer wird z.B. das erste Laufzeitsystem der SPS angesprochen. In dem Index-Group und Index-Offset Parameter wird der auszuführende Dienst verschlüsselt. Hier sollen 10 Byte Daten einer SPS-Variablen im Merkerbereich ab dem Byte-Offset 100 gelesen werden. Beim Erfolg werden 10 Byte Daten an die Adresse der *Buffer*-Variablen hineinkopiert.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.7.0 & TwinCAT v2.8.0	BC9xxx (165) firmware version >= 0xB6	TcAdsBC.Lb6

3.2 ADSRDWRTEX



Der Baustein erlaubt die Ausführung eines kombinierten ADS-Schreib-Lesebefehls. Mit einem Aufruf werden Daten zu einem ADS-Gerät übermittelt (Write) und dessen Antwortdaten gelesen (Read).

VAR_INPUT

```
VAR_INPUT
    NETID       : T_AmsNetId;
    PORT        : T_AmsPort;
    IDXGRP      : UDINT;
    IDXOFFS     : UDINT;
    WRITELEN    : UDINT;
    READLEN     : UDINT;
    SRCADDR     : DWORD;
    DESTADDR    : DWORD;
    WRTRD       : BOOL;
    TMOUT       : TIME;
END_VAR
```

T_AmsNetId [▶ 31]

T_AmsPort [▶ 31]

NETID : Ist ein String, der die AMS-Netzwerkkennung des Zielgerätes enthält, an das der ADS-Befehl gerichtet wird.

PORT : Enthält die Portnummer des ADS-Gerätes.

IDXGRP : Enthält die Index-Gruppennummer (32bit, unsigned) des angeforderten ADS-Dienstes. Dieser Wert ist der ADS-Tabelle des angesprochenen Gerätes zu entnehmen.

IDXOFFS : Enthält die Index-Offsetnummer (32bit, unsigned) des angeforderten ADS-Dienstes. Dieser Wert ist der ADS-Tabelle des angesprochenen Gerätes zu entnehmen.

WRITELEN : Enthält die Anzahl der zu schreibenden Daten in Bytes.

READLEN : Enthält die Anzahl der zu lesenden Daten in Bytes.

SRCADDR : Enthält die Adresse des Puffers, aus dem die zu schreibenden Daten geholt werden sollen. Der Programmierer ist selbst dafür verantwortlich, den Puffer in der Größe so zu dimensionieren, dass ‚WRITELEN'- Bytes daraus entnommen werden können. Der Puffer kann eine Einzelvariable, ein Array oder eine Struktur sein, dessen Adresse man mit dem ADR - Operator ermitteln kann.

DESTADDR : Enthält die Adresse des Puffers, der die gelesenen Daten aufnehmen soll. Der Programmierer ist selbst dafür verantwortlich den Puffer in der Größe so zu dimensionieren, dass er ‚READLEN' Bytes aufnehmen kann. Der Puffer kann eine Einzelvariable, ein Array oder eine Struktur sein, dessen Adresse man mit dem ADR - Operator ermitteln kann.

WRTRD : Durch eine steigende Flanke an diesem Eingang wird der ADS-Befehl ausgelöst.

TMOUT : Gibt die Zeit bis zum Abbrechen der Funktion an.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY        : BOOL;
    ERR         : BOOL;
    ERRID       : UDINT;
    COUNT_R     : UDINT;
END_VAR
```

BUSY : Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

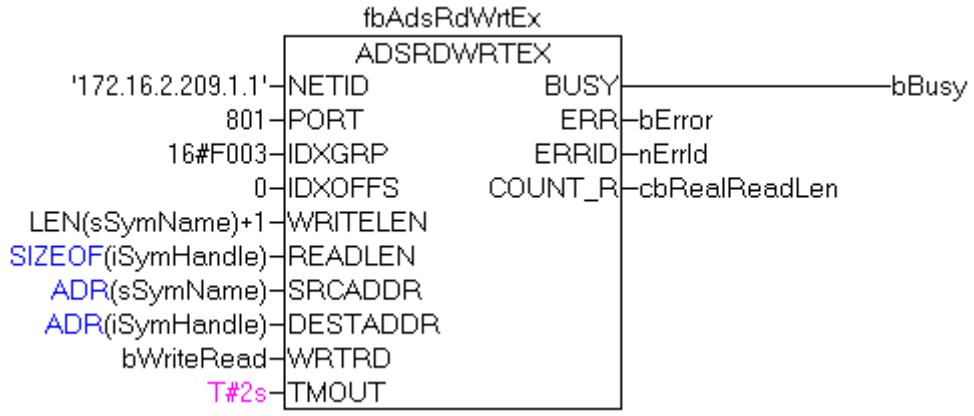
ERR : Sollte ein ADS-Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

ERRID : Liefert bei einem gesetzten ERR-Ausgang die ADS- oder die Gerätespezifische-Fehlernummer [▶ 33].

COUNT_R: Anzahl der erfolgreich gelesenen Datenbytes.

Beispiel für einen Aufruf in FUP:

```
PROGRAM MAIN
VAR
    fbAdsRdWrtEx : ADSRDWRTEX;
    sSymName     : STRING:='MAIN.VARCOUNTER';
    iSymHandle   : UDINT;
    bWriteRead   : BOOL;
    bBusy        : BOOL;
    bError       : BOOL;
    nErrId       : UDINT;
    cbRealReadLen: UDINT;
END_VAR
```

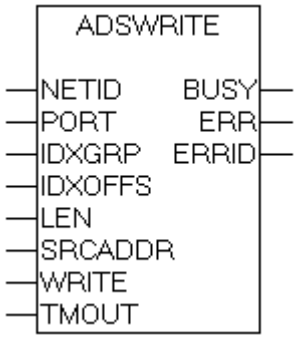


In dem Beispiel wird ein ADS-Kommando an ein ADS-Gerät mit der Netzwerkadresse: '172.16.2.209.1.1' und der Portnummer 801 gesendet. Über die Portnummer 801 kann z.B. das erste Laufzeitsystem der SPS angesprochen werden. Im Index-Group und Index-Offset wird der ADS-Dienst verschlüsselt. Hier soll das Handle einer SPS-Variablen mit dem Symbolnamen: 'MAIN.VARCOUNTER' gelesen und an den Aufrufer zurückgeliefert werden. Der WRITELEN-Eingangsparameter enthält die Stringlänge des Symbolnamens + 1 Byte für die Abschließende NULL. Beim Erfolg werden 4 Byte Daten an die Adresse der *iSymHandle*-Variablen hineinkopiert.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.7.0 & TwinCAT v2.8.0	BC9xxx (165) firmware version >= 0xB6	TcAdsBC.Lb6

3.3 ADSWRITE



Der Baustein erlaubt die Ausführung eines ADS-Schreibbefehls, um Daten zu einem ADS-Gerät zu übermitteln.

VAR_INPUT

```
VAR_INPUT
    NETID       : T_AmsNetId;
    PORT        : T_AmsPort;
    IDXGRP      : UDINT;
    IDXOFFS     : UDINT;
    LEN         : UDINT;
    SRCADDR     : DWORD;
    WRITE       : BOOL;
    TMOUT       : TIME;
END_VAR
```

T_AmsNetId [▶ 31]

T_AmsPort [▶ 31]

NETID : Ist ein String, der die AMS-Netzwerkkennung des Zielgerätes enthält, an das der ADS-Befehl gerichtet wird.

PORT : Enthält die Portnummer des ADS-Gerätes an das der Befehl gerichtet ist.

IDXGRP : Enthält die Index-Gruppennummer (32bit, unsigned) des angeforderten ADS-Dienstes. Dieser Wert ist der ADS-Tabelle des angesprochenen Gerätes zu entnehmen.

IDXOFFS : Enthält die Index-Offsetnummer (32bit, unsigned) des angeforderten ADS-Dienstes. Dieser Wert ist der ADS-Tabelle des angesprochenen Gerätes zu entnehmen.

LEN : Enthält die Anzahl der zu lesenden Daten in Bytes.

SRCADDR : Enthält die Adresse des Puffers, aus dem die zu schreibenden Daten geholt werden sollen. Der Programmierer ist selbst dafür verantwortlich, den Puffer in der Größe so zu dimensionieren, dass ‚LEN'-Bytes daraus entnommen werden können. Der Puffer kann eine Einzelvariable, ein Array oder eine Struktur sein, dessen Adresse man mit dem ADR - Operator ermitteln kann.

WRITE : Durch eine steigende Flanke an diesem Eingang wird der ADS-Befehl ausgelöst.

TMOUT : Gibt die Zeit bis zum Abbrechen der Funktion an.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY        : BOOL;
    ERR         : BOOL;
    ERRID       : UDINT;
END_VAR
```

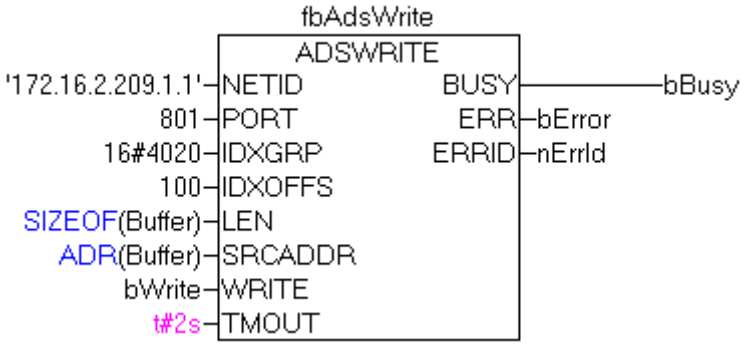
BUSY : Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

ERR : Sollte ein ADS-Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

ERRID : Liefert bei einem gesetzten ERR-Ausgang die ADS- oder die Gerätespezifische-Fehlernummer [▶ 33].

Beispiel für den Aufruf des Bausteins in FBD:

```
PROGRAM MAIN
VAR
    fbAdsWrite      : ADSWRITE;
    bWrite      : BOOL;
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
    Buffer      : ARRAY[1..10] OF BYTE := 1,2,3,4,5,6,7,8,9,0;
END_VAR
```

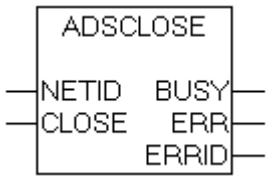


Im Beispiel wird ein ADS-Kommando an ein ADS-Gerät mit der Netzwerkadresse: '172.16.2.209.1.1' und der Portnummer 801 gesendet. Über diese Portnummer wird z.B. das erste Laufzeitsystem der SPS angesprochen. In dem Index-Group und Index-Offset Parameter wird der auszuführende Dienst verschlüsselt. Beim Erfolg werden 10 Byte Daten (*Buffer*-Variable) in den Merkerbereich ab dem Byte-Offset 100 des Zielgerätes geschrieben.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.7.0 & TwinCAT v2.8.0	BC9xxx (165) firmware version >= 0xB6	TcAdsBC.Lb6

3.4 ADSCLOSE



Mit dem Funktionsbaustein ADSCLOSE kann eine nicht benötigte IP-Verbindung explizit geschlossen werden. Um den Resourcenverbrauch minimal zu halten, ist die Anzahl der maximal gleichzeitig geöffneten IP-Verbindungen auf 4 begrenzt. Eine Verbindung wird beim Aufruf von ADSREADEX, ADSWRITE oder ADSRDWRTEX automatisch aufgebaut. Die nicht benutzte Verbindungen werden nach 10 Sekunden automatisch abgebaut . Soll in dieser Zeit eine Verbindung zu mehr als 4 ADS-Geräten aufgebaut werden, dann müssen die nicht benötigten Verbindungen zuerst geschlossen werden.

VAR_INPUT

```
VAR_INPUT
    NETID       : T_AmsNetId;
    CLOSE       : BOOL;
    TMOUT       : TIME;
END_VAR
```

T_AmsNetId [▶ 31]

NETID : Die AMS-Netzwerkkennung des ADS-Gerätes dessen Verbindung getrennt werden soll.

CLOSE : Durch eine steigende Flanke an diesem Eingang wird der ADS-Befehl ausgelöst.

TMOUT : Gibt die Zeit an, die bei der Ausführung des ADS-Kommandos nicht überschritten werden kann.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY        : BOOL;
    ERR         : BOOL;
    ERRID       : UDINT;
END_VAR
```

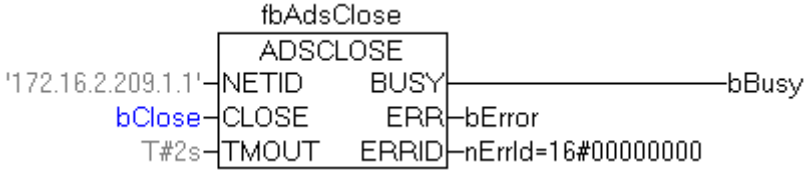
BUSY : Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

ERR : Sollte ein ADS-Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

ERRID : Liefert bei einem gesetzten ERR-Ausgang die ADS- oder die Gerätespezifische-Fehlernummer [▶ 33].

Beispiel für den Aufruf des Bausteins in FBD:

```
PROGRAM MAIN
VAR
    fbAdsClose  : ADSCLOSE;
    bClose      : BOOL;
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
END_VAR
```



Im Beispiel wird bei einer steigender Flanke am *bClose*-Eingang die IP-Verbindung zu dem ADS-Gerät mit der Netzwerkadresse: '172.16.2.209.1.1' abgebaut.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.7.0 & TwinCAT v2.8	BC9xxx (165) firmware version >= 0xB6	TcAdsBC.Lb6

3.5 ADS Indication/Response

Die ADS Indication-/Response-Funktionsbausteine ermöglichen den Aufbau einer Client-Server-Kommunikation zwischen einem ADS-Gerät und einer SPS-Task eines Buscontrollers oder zwischen zwei Buscontrollern (z.B. BC9000). Bei dem ADS-Gerät kann sich z.B. um eine Windows-Applikation (nutzt die AdsDLL/AdsOcx) oder ein anderes SPS-Laufzeitsystem handeln. Die Kommunikation zwischen dem ADS-Gerät und der SPS-Task wird mittels folgenden Dienstprimitiven abgewickelt:

- Request
- Indication
- Response
- Confirmation

Die Kommunikation zwischen einem ADS-Gerät und einer SPS-Task hat folgenden Ablauf: Ein ADS-Gerät sendet ein Request (Anfrage) an das Zielgerät (SPS-Task). Diese Anfrage wird durch eine Indication in dem Zielgerät registriert. Das Zielgerät (SPS-Task) führt darauf einen entsprechenden Dienst aus. Der auszuführende Dienst wird über die Index-Group/Offset Parameter verschlüsselt. Danach sendet die SPS-Task ein Response (Antwort) an das ADS-Gerät. Das Response wird von dem ADS-Quellgerät als Confirmation registriert.
Pro SPS-Task kann sinnvoll nur eine Instanz des Indication- und Response-Funktionsbausteins benutzt werden. Entsprechend den verfügbaren ADS-Diensten: READ, WRITE und READ & WRITE gibt es zu jedem Dienst einen entsprechenden Indication bzw. Response Funktionsbaustein.

Die ADS-Geräte werden über eine Portadresse (PORT) und eine Netzwerkadresse (NETID) adressiert.

Beispiel:

Die SPS-Task eines Buscontrollers BC9000 mit der Netzwerkadresse "172.64.23.12.1.1" soll angesprochen werden. Die SPS-Task des Buscontrollers besitzt die Portnummer: 800.

Die Netzwerkadresse:
PORT = 800
NETID = '172.64.23.12.1.1'

Bemerkungen:

- **Damit ein Request an die SPS-Task weitergeleitet wird, muss in dem IndexGroup-Parameter beim Request das höchstwertige Bit gesetzt werden z.B. IG:=0x80000001.**
- **Die maximale Datengröße, die ein Buscontroller BC9xxx senden (z.B. mit ADSREADRESBC) und empfangen kann (z.B. mit ADSWRITEIND) ist auf 1980 Byte begrenzt.**

Tab. 1: ADS Indication-/Response-Funktionsbausteine

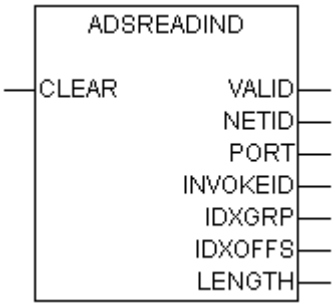
Dienst	Name	Beschreibung
READ	ADSREADIND [▶ 17]	ADSREAD-Indication.

Dienst	Name	Beschreibung
	ADSREADRESBC [▶ 21]	ADSREAD-Response
WRITE	ADSWRITEIND [▶ 18]	ADSWRITE-Indication
	ADSWRITERESBC [▶ 22]	ADSWRITE-Response
READ & WRITE	ADSRDWRTIND [▶ 20]	ADS-READ & WRITE-Indication
	ADSRDWRTRESBC [▶ 23]	ADS-READ & WRITE-Response

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.7.0 Build > 517 TwinCAT v2.8.0 Build > 729	BC9xxx (165) firmware version >= 0xB7	TcAdsBC.Lb6

3.5.1 ADSREADIND



Der Funktionsbaustein registriert ADSREAD-Anfragen (ADSREAD-Requests) an eine SPS-Task als Indications und erlaubt deren Bearbeitung. Das Anstehen einer Indication wird über eine steigende Flanke am VALID-Ausgang gemeldet. Über eine positive Flanke am CLEAR-Eingang wird die Indication als bearbeitet gemeldet. Eine negative Flanke am CLEAR-Eingang gibt den Funktionsbaustein für die Verarbeitung weiterer Indications frei. Nachdem eine Indication bearbeitet wurde, muss eine Antwort über den ADSREADRESBC [▶ 21]-Funktionsbaustein an das Quellgerät gesendet werden. Die Parameter: PORT, NETID können dafür benutzt werden um das Quellgerät zu adressieren. Der INVOKEID-Parameter dient dem Quellgerät der Zuordnung der Antworten zu den Anfragen und wird ebenfalls als Parameter an das Quellgerät zurück gesendet.

VAR_INPUT

```
VAR_INPUT
    CLEAR       : BOOL;
END_VAR
```

CLEAR : Mit einer steigenden Flanke an diesem Eingang wird eine Indication als bearbeitet gemeldet und die Ausgänge des ADSREADIND-Funktionsbausteins zurückgesetzt. Eine fallende Flanke gibt den Funktionsbaustein für die Verarbeitung weiterer Indications frei.

VAR_OUTPUT

```
VAR_OUTPUT
    VALID       : BOOL;
    NETID       : T_AmsNetId;
    PORT        : T_AmsPort;
    INVOKEID        : UDINT;
    IDXGRP      : UDINT;
    IDXOFFS     : UDINT;
    LENGTH      : UDINT;
END_VAR
```

VALID : Der Ausgang ist gesetzt, wenn von dem Funktionsbaustein eine Indication registriert wurde und bleibt gesetzt bis diese über eine positive Flanke an dem CLEAR-Eingang als bearbeitet gemeldet wurde.

NETID : Ist ein String, der die AMS-Netzwerkkennung des Quellgerätes enthält, von dem der ADS-Befehl gesendet wurde.

PORT : Enthält die Portnummer des ADS-Quellgerätes von dem der ADS-Befehl gesendet wurde.

INVOKEID : Enthält ein Handle des Befehls, der gesendet wurde. Die InvokeId wird von dem Quellgerät festgelegt und dient der Identifizierung der Befehle.

IDXGRP : Enthält die Index-Gruppennummer (32bit, unsigned) des angeforderten ADS-Dienstes.

IDXOFFS : Enthält die Index-Offset-Nummer (32bit, unsigned) des angeforderten ADS-Dienstes.

LENGTH : Enthält die Anzahl der zu lesenden Daten in Bytes.

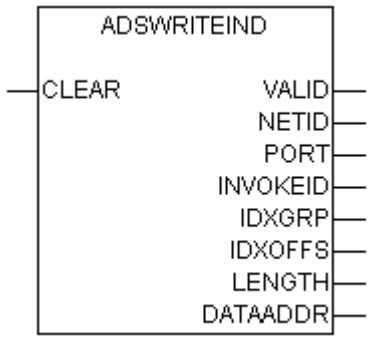
Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.7.0 Build > 517 TwinCAT v2.8.0 Build > 729	BC9xxx (165) firmware version >= 0xB7	TcAdsBC.Lb6

Sehen Sie dazu auch

- T_AmsNetId [▶ 31]
- T_AmsPort [▶ 31]

3.5.2 ADSWRITEIND



Der Funktionsbaustein registriert ADSWRITE-Anfragen (ADSWRITE-Requests) an eine SPS-Task als Indications und erlaubt deren Bearbeitung. Das Anstehen einer Indication wird über eine steigende Flanke am VALID-Ausgang gemeldet. Über eine positive Flanke am CLEAR-Eingang wird die Indication als bearbeitet gemeldet. Eine fallende Flanke am CLEAR-Eingang gibt den Funktionsbaustein für die Verarbeitung weiterer Indications frei. Nachdem eine Indication bearbeitet wurde, muss eine Antwort über den ADSWRITERESBC [▶ 22]-Funktionsbaustein an das Quellgerät gesendet werden. Die Parameter: PORT,

NETID können dafür benutzt werden, um das Quellgerät zu adressieren. Der INVOKEID-Parameter dient dem Quellgerät der Zuordnung der Antworten zu den Anfragen und wird ebenfalls als Parameter an das Quellgerät zurückgesendet.

VAR_INPUT

```
VAR_INPUT
    CLEAR       : BOOL;
END_VAR
```

CLEAR: Mit einer steigenden Flanke an diesem Eingang wird eine Indication als bearbeitet gemeldet und die Ausgänge des ADSWRITEIND-Funktionsbausteins zurückgesetzt (DATAADDR = 0, LENGTH = 0 !). Eine fallende Flanke gibt den Funktionsbaustein für die Verarbeitung weiterer Indications frei.

VAR_OUTPUT

```
VAR_OUTPUT
    VALID       : BOOL;
    NETID       : T_AmsNetId;
    PORT        : T_AmsPort;
    INVOKEID        : UDINT;
    IDXGRP      : UDINT;
    IDXOFFS     : UDINT;
    LENGTH      : UDINT;
    DATAADDR        : DWORD;
END_VAR
```

VALID : Der Ausgang ist gesetzt, wenn von dem Funktionsbaustein eine Indication registriert wurde und bleibt gesetzt bis diese über eine positive Flanke an dem CLEAR-Eingang als bearbeitet gemeldet wurde.

NETID : Ist ein String, der die AMS-Netzwerkkennung des Quellgerätes enthält, von dem der ADS-Befehl gesendet wurde.

PORT : Enthält die Portnummer des ADS-Quellgerätes von dem der ADS-Befehl gesendet wurde.

INVOKEID : Enthält ein Handle des Befehls, der gesendet wurde. Die InvokeId wird von dem Quellgerät festgelegt und dient der Identifizierung der Befehle.

IDXGRP : Enthält die Index-Gruppennummer (32bit, unsigned) des angeforderten ADS-Dienstes.

IDXOFFS : Enthält die Index-Offset-Nummer (32bit, unsigned) des angeforderten ADS-Dienstes.

LENGTH : Enthält die Länge der geschriebenen Daten in Bytes.

DATAADDR : Enthält die Adresse des Datenpuffers in dem sich die geschriebenen Daten befinden.

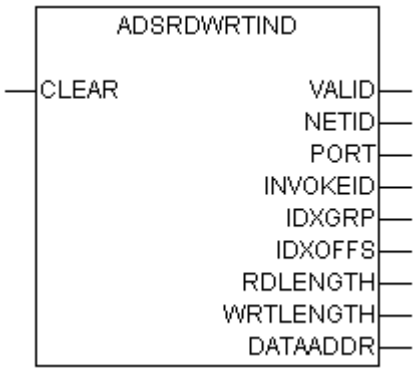
Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.7.0 Build > 517 TwinCAT v2.8.0 Build > 729	BC9xxx (165) firmware version >= 0xB7	TcAdsBC.Lb6

Sehen Sie dazu auch

- T_AmsNetId [▶ 31]
- T_AmsPort [▶ 31]

3.5.3 ADSRDWRTIND



Der Funktionsbaustein registriert ADSRDWRT-Anfragen (ADSRDWRT-Requests) an eine SPS-Task als Indications und erlaubt deren Bearbeitung. Das Anstehen einer Indication wird über eine steigende Flanke am VALID-Ausgang gemeldet. Über eine positive Flanke am CLEAR-Eingang wird die Indication als bearbeitet gemeldet. Eine fallende Flanke am CLEAR-Eingang gibt den Funktionsbaustein für die Verarbeitung weiterer Indications frei. Nachdem eine Indication bearbeitet wurde, muss eine Antwort über den ADSRDWRTRESBC [▶ 23]-Funktionsbaustein an das Quellgerät gesendet werden. Die Parameter: PORT, NETID können dafür benutzt werden um das Quellgerät zu adressieren. Der INVOKEID-Parameter dient dem Quellgerät der Zuordnung der Antworten zu den Anfragen und wird ebenfalls als Parameter an das Quellgerät zurück gesendet.

VAR_INPUT

```
VAR_INPUT
    CLEAR       : BOOL;
END_VAR
```

CLEAR : Mit einer steigenden Flanke an diesem Eingang wird eine Indication als bearbeitet gemeldet und die Ausgänge des ADSRDWRTIND-Funktionsbausteins zurückgesetzt. Eine fallende Flanke gibt den Funktionsbaustein für die Verarbeitung weiterer Indications frei.

VAR_OUTPUT

```
VAR_OUTPUT
    VALID       : BOOL;
    NETID       : T_AmsNetId;
    PORT        : T_AmsPort;
    INVOKEID        : UDINT;
    IDXGRP      : UDINT;
    IDXOFFS     : UDINT;
    RDLENGTH        : UDINT;
    WRTLENGTH           : UDINT;
    DATAADDR        : DWORD;
END_VAR
```

VALID : Der Ausgang ist gesetzt, wenn von dem Funktionsbaustein eine Indication registriert wurde und bleibt gesetzt bis diese über eine positive Flanke an dem CLEAR-Eingang als bearbeitet gemeldet wurde.

NETID : Ist ein String, der die AMS-Netzwerkkennung des Quellgerätes enthält, von dem der ADS-Befehl gesendet wurde.

PORT : Enthält die Portnummer des ADS-Quellgerätes von dem der ADS-Befehl gesendet wurde.

INVOKEID : Enthält ein Handle des Befehls, der gesendet wurde. Die InvokeId wird von dem Quellgerät festgelegt und dient der Identifizierung der Befehle.

IDXGRP : Enthält die Index-Gruppennummer (32bit, unsigned) des angeforderten ADS-Dienstes.

IDXOFFS : Enthält die Index-Offset-Nummer (32bit, unsigned) des angeforderten ADS-Dienstes.

RDLENGTH : Enthält die Länge der zu lesenden Daten in Bytes.

WRTLENGTH : Enthält die Länge der geschriebenen Daten in Bytes.

DATAADDR : Enthält die Adresse des Datenpuffers in dem sich die geschriebenen Daten befinden.

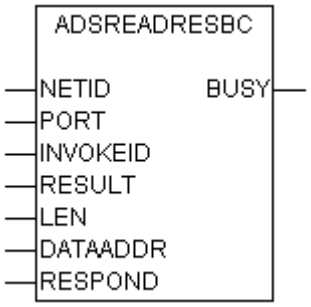
Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.7.0 Build > 517 TwinCAT v2.8.0 Build > 729	BC9xxx (165) firmware version >= 0xB7	TcAdsBC.Lb6

Sehen Sie dazu auch

- T_AmsNetId [▶ 31]
- T_AmsPort [▶ 31]

3.5.4 ADSREADRESBC



Der Funktionsbaustein ADSREADRESBC wird dafür benutzt um READ-Indications einer SPS-Task zu quittieren. Über eine positive Flanke am RESPOND-Eingang wird eine Antwort an das ADS-Quellgerät gesendet. Das Quellgerät wird über die Parameter: PORT und NETID adressiert. Der Parameter INVOKEID dient dem Quellgerät der Zuordnung der Antworten zu den Anfragen und wird von dem Ausgang des ADSREADIND [▶ 17]-Funktionsbaustein übernommen. Über den RESULT-Parameter kann ein Fehlercode an das ADS-Quellgerät zurückgegeben werden.

VAR_INPUT

```
VAR_INPUT
    NETID       : T_AmsNetId;
    PORT        : T_AmsPort;
    INVOKEID        : UDINT;
    RESULT      : UDINT;
    LEN         : UDINT;
    DATAADDR        : DWORD;
    RESPOND     : BOOL;
END_VAR
```

NETID : Ist ein String, der die AMS-Netzwerkkennung des Quellgerätes enthält, an den der ADS-Befehl gesendet werden soll.

PORT : Enthält die Portnummer des ADS-Quellgerätes, an den die Antwort gesendet werden soll

INVOKEID : Enthält ein Handle des Befehls, der gesendet wurde. Die InvokeId wird von dem Quellgerät festgelegt und dient der Identifizierung der Befehle.

RESULT : Enthält den Fehlercode, der an das Quellgerät gesendet werden soll

LEN : Enthält die Anzahl der zu lesenden Daten in Bytes.

DATAADDR : Enthält die Adresse des Datenpuffers, der gelesen werden sollte.

RESPOND : Über eine positive Flanke an diesem Eingang wird der Funktionsbaustein aktiviert.

VAR_OUTPUT

```
VAR_INPUT
    BUSY        : BOOL;
END_VAR
```

BUSY : Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

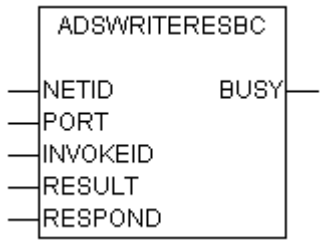
Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.7.0 Build > 517 TwinCAT v2.8.0 Build > 729	BC9xxx (165) firmware version >= 0xB7	TcAdsBC.Lb6

Sehen Sie dazu auch

- T_AmsNetId [▶ 31]
- T_AmsPort [▶ 31]

3.5.5 ADSWRITERESBC



Der Funktionsbaustein ADSWRITERESBC wird dafür benutzt um Indications einer SPS-Task zu quittieren. Über eine positive Flanke am RESPOND-Eingang wird eine Antwort an das ADS-Quellgerät gesendet. Das Quellgerät wird über die Parameter: PORT und NETID adressiert. Der Parameter INVOKEID dient dem Quellgerät der Zuordnung der Antworten zu den Anfragen und wird von dem Ausgang des ADSWRITEIND [▶ 17]-Funktionsbausteins übernommen. Über den RESULT-Parameter kann ein Fehlercode an das ADS-Quellgerät zurückgegeben werden.

VAR_INPUT

```
VAR_INPUT
    NETID       : T_AmsNetId;
    PORT        : T_AmsPort;
    INVOKEID        : UDINT;
```

```
    RESULT      : UDINT;
    RESPOND     : BOOL;
END_VAR
```

NETID : Ist ein String, der die AMS-Netzwerkkennung des Quellgerätes enthält, an den der ADS-Befehl gesendet werden soll.

PORT : Enthält die Portnummer des ADS-Quellgerätes an den der ADS-Befehl gesendet werden soll

INVOKEID : Enthält ein Handle des Befehls, der gesendet wurde. Die InvokeId wird von dem Quellgerät festgelegt und dient der Identifizierung der Befehle.

RESULT : Enthält den Fehlercode, der an das Quellgerät gesendet werden soll

RESPOND : Über eine positive Flanke an diesem Eingang wird der Funktionsbaustein aktiviert.

VAR_OUTPUT

```
VAR_INPUT
    BUSY        : BOOL;
END_VAR
```

BUSY : Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

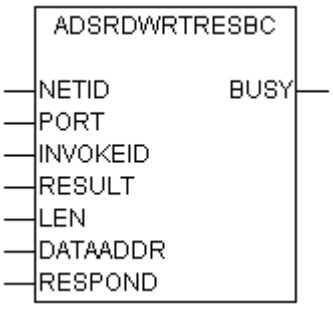
Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.7.0 Build > 517 TwinCAT v2.8.0 Build > 729	BC9xxx (165) firmware version >= 0xB7	TcAdsBC.Lb6

Sehen Sie dazu auch

- T_AmsNetId [▶ 31]
- T_AmsPort [▶ 31]

3.5.6 ADSRDWRTRESBC



Der Funktionsbaustein ADSRDWRTRESBC wird dafür benutzt um Indications einer SPS-Task zu quittieren. Über eine positive Flanke am RESPOND-Eingang wird eine Antwort an das ADS-Quellgerät gesendet. Das Quellgerät wird über die Parameter: PORT und NETID adressiert. Der Parameter INVOKEID dient dem Quellgerät der Zuordnung der Antworten zu den Anfragen und wird von dem Ausgang des ADSRDWRTIND [▶ 20]-Funktionsbausteins übernommen. Über den RESULT-Parameter kann ein Fehlercode an das ADS-Quellgerät zurückgegeben werden.

VAR_INPUT

```
VAR_INPUT
    NETID       : T_AmsNetId;
    PORT        : T_AmsPort;
    INVOKEID        : UDINT;
    RESULT      : UDINT;
    LEN         : UDINT;
    DATAADDR        : DWORD;
    RESPOND     : BOOL;
END_VAR
```

NETID : Ist ein String, der die AMS-Netzwerkkennung des Quellgerätes enthält, an den der ADS-Befehl gesendet werden soll.

PORT : Enthält die Portnummer des ADS-Quellgerätes an den der ADS-Befehl gesendet werden soll.

INVOKEID : Enthält ein Handle des Befehls, der gesendet wurde. Die InvokeId wird von dem Quellgerät festgelegt und dient der Identifizierung der Befehle.

RESULT : Enthält den Fehlercode, der an das Quellgerät gesendet werden soll.

LEN : Länge der gelesenen Daten in Byte. Diese Daten werden an das Quellgerät zurückgesendet.

DATAADDR : Adresse des Datenpuffers in dem sich die gelesenen Daten befinden.

RESPOND : Über eine positive Flanke an diesem Eingang wird der Funktionsbaustein aktiviert.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY        : BOOL;
END_VAR
```

BUSY : Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.7.0 Build > 517 TwinCAT v2.8.0 Build > 729	BC9xxx (165) firmware version >= 0xB7	TcAdsBC.Lb6

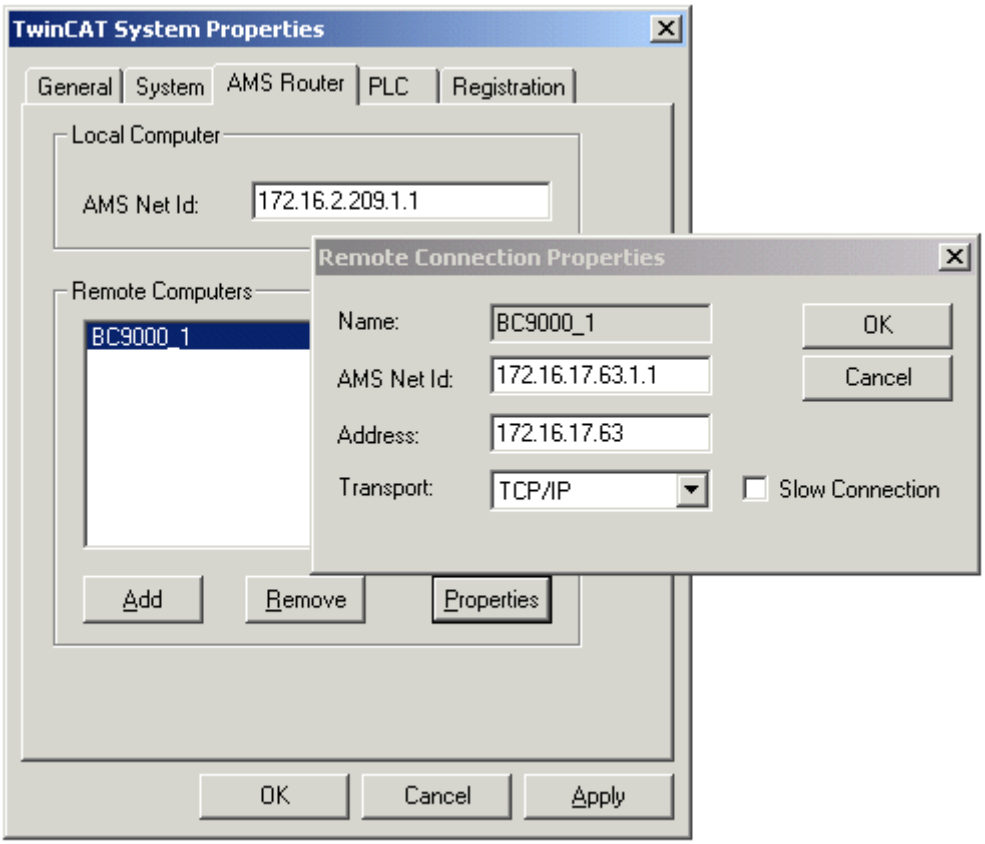
Sehen Sie dazu auch

- T_AmsNetId [▶ 31]
- T_AmsPort [▶ 31]

3.5.7 Beispiel 1: ADSREAD-Indication/-Response

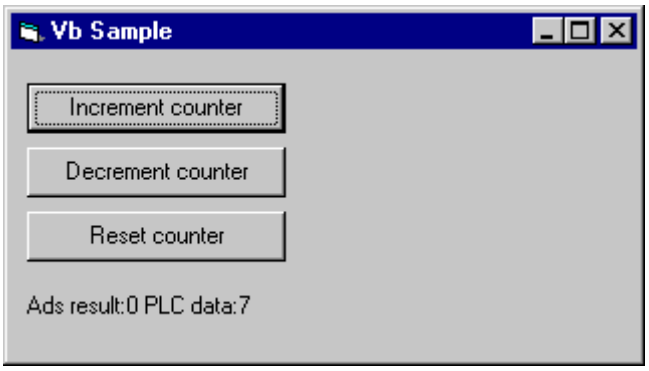
In der Beispielapplikation werden von einer VB-Applikation READ-Requests an die SPS-Task eines BC9000 gesendet um eine SPS-Zählervariable zu inkrementieren/dekrementieren oder zurückzusetzen. Die READ-Requests werden bei dem Buscontroller als Indications abgefangen und die gewünschte Operation an der Zählervariable ausgeführt. Danach wird von dem Buscontroller ein Response mit dem aktuellen Wert der Zählervariable an die VB-Applikation zurück gesendet. Der Wert der Zählervariable wird auf der Form ausgegeben. Um mit der SPS-Task zu kommunizieren benutzt die VB-Applikation das ActiveX Control: AdsOcx. Der Buscontroller muss als Remote-PC im TwinCAT-System-Menü eingetragen werden um die Anfragen der VB-Applikation zu dem Buscontroller übers Ethernet routen zu können. Hier können Sie die kompletten https://infosys.beckhoff.com/content/1031/tcplclibadsbc/Resources/12261725067.exe.

BC9000 als Remote-PC im TwinCAT System Menü eintragen



Das TwinCAT Systemmenü kann mit einem Mausklick auf das TwinCAT Icon -> Eigenschaften in der Taskleiste erreicht werden. Auf dem *AMS Router*-Reiter wählen Sie: *Add.* In dem Dialog geben Sie einen beliebigen Namen für den Remote-PC ein. Unter *AMS Net Id* wird die Netzwerkadresse eingetragen. Diese setzt sich aus der IP-Adresse des BC9000 und zwei weiteren Stellen zusammen.:"1.1". Unter *Address* wird noch mal die IP-Adresse des BC9000 eingetragen. Die IP-Adresse muss der Adresse entsprechen, die mit den DIP-Schaltern auf dem Buscontroller eingestellt wurde.

Die VB-Applikation



In der *Form_Load*-Routine wird eine Verbindung zu der SPS-Task des BC9000 aufgebaut. Der gewünschte Dienst in der SPS-Task wird in dem Index-Group-Parameter verschlüsselt:

- IG:0x80000001 -> Die Zählervariable inkrementieren;
- IG:0x80000002 -> Die Zählervariable dekrementieren;
- IG:0x80000003 -> Die Zählervariable = 0 setzen;

- IO: 0x0

Damit die Requests an die SPS-Task weiter geleitet werden, muss in dem Index-Group-Parameter das höchstwertige Bit gesetzt werden. Die Werte der Index-Group und Index-Offset Parameter können frei definiert werden. In unserem Beispiel erkennt der Buscontroller anhand der Index-Group und Index-Offset-Parameter, ob er die SPS-Variable inkrementieren/dekrementieren oder zurücksetzen soll.

```
Option Explicit

Dim tmpData(1) As Integer
Dim AdsResult    As Integer

Private Sub Command1_Click()

    AdsResult = AdsOcx1.AdsSyncReadReq(&H80000001, &H0, 2, tmpData)
    Label1.Caption = "Ads result:" & AdsResult & " PLC data:" & tmpData(0)

End Sub

Private Sub Command2_Click()
    AdsResult = AdsOcx1.AdsSyncReadReq(&H80000002, &H0, 2, tmpData)
    Label1.Caption = "Ads result:" & AdsResult & " PLC data:" & tmpData(0)

End Sub

Private Sub Command3_Click()
    AdsResult = AdsOcx1.AdsSyncReadReq(&H80000003, &H0, 2, tmpData)
    Label1.Caption = "Ads result:" & AdsResult & " PLC data:" & tmpData(0)

End Sub

Private Sub Form_Load()
    AdsOcx1.AdsAmsServerNetId = "172.16.17.63.1.1"
    AdsOcx1.AdsAmsServerPort = 800 'PLC task number'
End Sub
```

Das SPS Programm

Erstellen Sie ein neues SPS-Projekt für den Buscontroller. Unter Bibliotheksverwaltung müssen folgende SPS-Bibliotheken für den Buscontroller eingebunden werden: Standard.lb6, PlcHelperBC.lb6 und TcAdsBC.lb6.

Die Requests werden in der SPS-Task von einer Instanz des ADSREADIND [▶ 17]-Funktionsbausteins als Indications abgefangen. Danach werden die Parameter Index-Group, Index-Offset und die angeforderte Datenlänge auf Gültigkeit überprüft. In der CASE-Anweisung wird die gewünschte Operation an der SPS-Variablen durchgeführt. Beim Erfolg wird ein Response von einer Instanz des ADSREADRESBC [▶ 21]-Funktionsbausteins an den Aufrufer mit dem aktuellen Wert der SPS-Variablen zurückgesendet. Im Fehlerfall eine entsprechende Fehlermeldung. Zum Schluss werden die Flags CLEAR und RESPOND rückgesetzt um weitere Indications verarbeiten zu können.

```
PROGRAM MAIN
VAR
    fbREADIND                   : ADSREADIND;
    fbREADRESBC             : ADSREADRESBC;

    szNetId             : STRING(23);
    Port                : UINT;
    InvokeId                : UDINT;
    idxGrp              : UDINT;
    idxOffs             : UDINT;
    cbLength                : UDINT;
    ErrorNumber             : UDINT;

    varCounter              : INT;
END_VAR
```

```
fbREADIND( );
fbREADRESBC( );
```

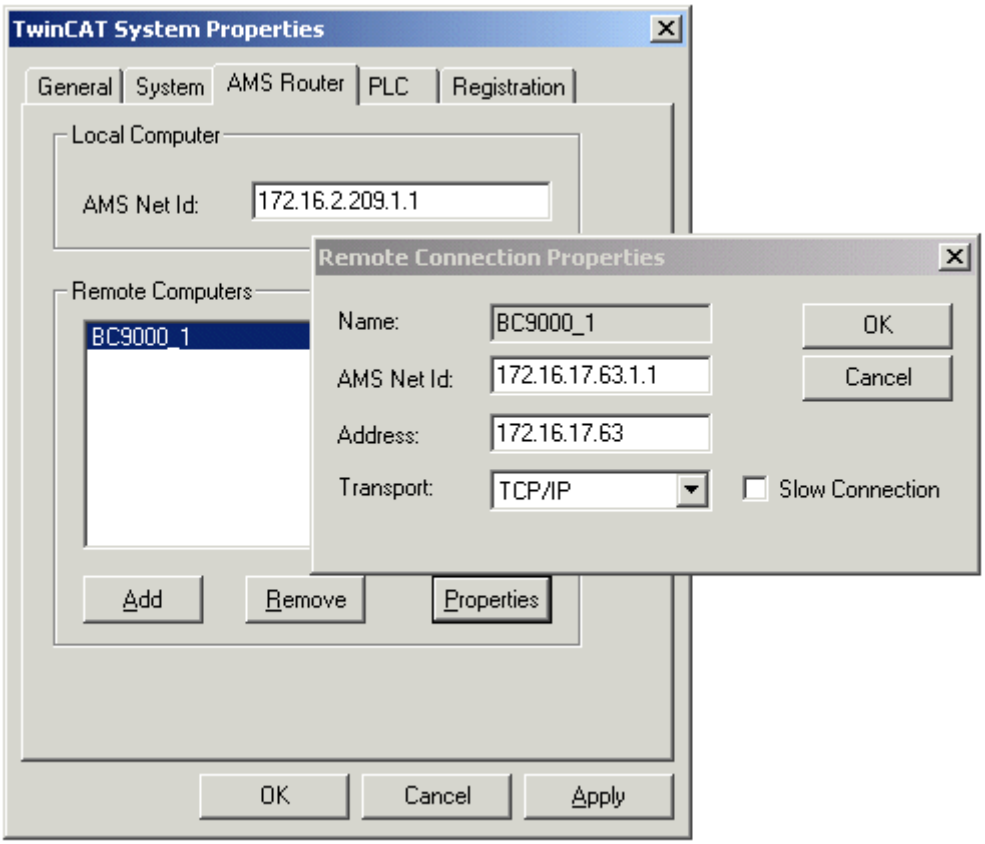
```
IF ( fbREADIND.VALID ) THEN
    szNetId := fbREADIND.NETID;
    Port := fbREADIND.PORT;
    InvokeId := fbREADIND.INVOKEID;
    idxGrp := fbREADIND.IDXGRP;
    idxOffs := fbREADIND.IDXOFFS;
    cbLength := fbREADIND.LENGTH;
    fbREADIND(  CLEAR := TRUE );        (*clear indication entry*)
    ErrorNumber := 0;

    IF idxOffs = 0 THEN
        IF  cbLength >= 2 THEN
            CASE idxGrp OF
                16#80000001:
                    varCounter := varCounter  + 1;

                16#80000002:
                    varCounter := varCounter  - 1;

                16#80000003:
                    varCounter := 0;
            ELSE
                ErrorNumber := 1793;  (* ADS error: Service is not supported by server*)
            END_CASE
        ELSE
            ErrorNumber := 1797;  (*ADS error:Parameter size not correct*)
        END_IF
    ELSE
        ErrorNumber := 1795;    (*ADS error: Invalid index offset*)
    END_IF

    fbREADRESBC.NETID := szNetId;
    fbREADRESBC.PORT := Port;
    fbREADRESBC.INVOKEID := InvokeId;
    fbREADRESBC.RESULT := ErrorNumber;

    IF ErrorNumber = 0 THEN
        fbREADRESBC( LEN := SIZEOF(varCounter),DATAADDR := ADR(varCounter), RESPOND := TRUE );
    ELSE
        fbREADRESBC( LEN := 0,DATAADDR :=0, RESPOND := TRUE );
    END_IF

END_IF

(*reset fb's*)
IF NOT fbREADRESBC.BUSY THEN
    fbREADIND( CLEAR := FALSE );
    fbREADRESBC( RESPOND := FALSE );
END_IF
```

Hier können Sie die kompletten Sourcen zu der Beispielapplikation entpacken: https://infosys.beckhoff.com/content/1031/tcplclibadsbc/Resources/12261725067.exe

3.5.8 Beispiel 2: ADSWRITE-Indication/-Response

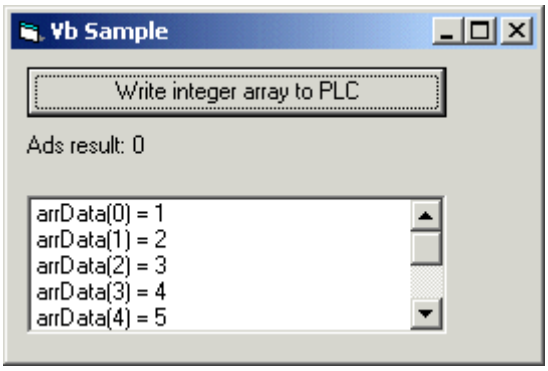
Die VB-Applikation sendet ein Array mit 10 Integer-Werten zu der SPS-Task eines Buscontrollers BC9000. In der SPS-Task des Buscontrollers sollen die empfangenen Daten in eine Array-Variable zur weiteren Verarbeitung kopiert werden. Um mit dem Buscontroller kommunizieren zu können, benutzt die VB-Applikation das ActiveX Control: AdsOcx. Der Buscontroller muss als Remote-PC im TwinCAT-System-Menü eingetragen werden um die Anfragen der VB-Applikation zu dem Buscontroller übers Ethernet routen zu können. Hier können Sie die kompletten https://infosys.beckhoff.com/content/1031/tcplclibadsbc/Resources/12261726475.exe.

BC9000 als Remote-PC im TwinCAT System Menü eintragen



Das TwinCAT Systemmenü kann mit einem Mausklick auf das TwinCAT Icon -> Eigenschaften in der Taskleiste erreicht werden. Auf dem *AMS Router*-Reiter wählen Sie *Add.* In dem Dialog geben Sie einen beliebigen Namen für den Remote-PC ein. Unter *AMS Net Id* wird die Netzwerkadresse eingetragen. Diese setzt sich aus der IP-Adresse des BC9000 und zwei weiteren Stellen zusammen.:"1.1". Unter *Address* wird noch mal die IP-Adresse des BC9000 eingetragen. Die IP-Adresse muss der Adresse entsprechen, die mit den DIP-Schaltern auf dem Buscontroller eingestellt wurde.

Die VB-Applikation



In der *Form_Load*-Routine wird eine Verbindung zu der SPS-Task des Buscontrollers BC9000 aufgebaut (Port 800 und Netzwerkadresse: "172.16.17.63.1.1"). Die ersten vier Stellen der Netzwerkadresse entsprechen der IP-Adresse des Buscontrollers. Der gewünschte Dienst in der SPS-Task wird in dem Index-Group und Index-Offset Parameter verschlüsselt. Z.B.:

- IG:0x80000005 und
- IO:0x00000007-> Die gesendeten Daten in den Array in der SPS kopieren.

Die VB-Applikation benutzt die Methode *AdsSyncWriteReq* um 20 Byte Daten an den Buscontroller zu senden. Die zuletzt gesendeten Werte werden zur Kontrolle einer Liste hinzugefügt.

Damit die Requests an die SPS-Task weiter geleitet werden, muss in dem Index-Group-Parameter das höchstwertige Bit gesetzt werden. Die Werte der Index-Group und Index-Offset Parameter können frei definiert werden. In unserem Beispiel erkennt der Buscontroller anhand der Index-Group und Index-Offset-Parameter, dass er die empfangenen Daten in ein Array kopieren soll.

```
Option Explicit

Dim AdsResult     As Integer
Dim arrData(0 To 9)   As Integer

Private Sub cmdWrite_Click()
    Call List1.Clear

    Dim i As Long
    For i = LBound(arrData) To UBound(arrData)
    arrData(i) = arrData(i) + 1 'change values
    Call List1.AddItem("arrData(" & i & ") = " & arrData(i))
    Next i

    'calculate the byte length parameter
    Dim cbWriteSize As Long
    cbWriteSize = (UBound(arrData) - LBound(arrData) + 1) * LenB(arrData(LBound(arrData)))

    AdsResult = AdsOcx1.AdsSyncWriteReq(&H80000005, &H7, cbWriteSize, arrData) 'send data to PLC
    Label1.Caption = "Ads result: " & AdsResult

End Sub

Private Sub Form_Load()
    AdsOcx1.AdsAmsServerNetId =  "172.16.17.63.1.1"
    AdsOcx1.AdsAmsServerPort = 800 'PLC task number of the BC9000 buscontroller'

    Dim i As Long
    For i = LBound(arrData) To UBound(arrData)
    arrData(i) = i    'init data
    Next i
End Sub
```

Das SPS Programm

Erstellen Sie ein neues SPS-Projekt für den Buscontroller. Unter Bibliotheksverwaltung müssen folgende SPS-Bibliotheken für den Buscontroller eingebunden werden: Standard.lb6, PlcHelperBC.lb6 und TcAdsBC.lb6.

In der SPS-Task des Buscontrollers wird eine Instanz von dem ADSWRITEIND [▶ 18]-Funktionsbaustein benutzt um die Daten zu empfangen und eine Instanz von dem ADSWRITERESBC [▶ 22]-Funkstionbaustein um den Empfang der Daten zu quittieren. Die Anfragen werden in der SPS-Task von dem ADSWRITEIND-Funktionsbaustein als sogenannte Indications abgefangen. Danach werden die Parameter Index-Group, Index-Offset und die gesendete Datenlänge auf Gültigkeit überprüft und die empfangenden Daten in eine Array-Variable kopiert. Als Nächstes wird ein Response von einer Instanz des ADSWRITERESBC-Funktionsbausteins an den Aufrufer (eventuell mit einem Fehlercode) zurückgesendet. Damit wird der VB-Applikation mitgeteilt, daß die gesendeten Daten erfolgreich empfangen wurden.

Mit der steigenden Flanke am CLEAR-Eingang des ADSWRITEIND-Funktionsbausteins wird der Adresspointer auf die zuletzt gesendeten Daten ungültig (== NULL). Aus diesem Grund werden die gesendeten Daten zuerst in die SPS-Variable kopiert und dann der CLEAR-Eingang auf TRUE gesetzt. Zum Schluss werden die Flags CLEAR und RESPOND rückgesetzt, um weitere Indications verarbeiten zu können.

```
PROGRAM MAIN
VAR
    fbWRITEIND              : ADSWRITEIND;
    fbWRITERESBC            : ADSWRITERESBC;

    szNetId             : STRING(23);
    Port                : UINT;
    InvokeId                : UDINT;
    idxGrp              : UDINT;
    idxOffs             : UDINT;
    cbLength                : UDINT;
```

```
    ErrorNumber             : UDINT;

    arrInt              : ARRAY[0..9] OF INT;
END_VAR
```

```
fbWRITEIND( );
fbWRITERESBC( );

IF ( fbWRITEIND.VALID ) THEN
    szNetId         := fbWRITEIND.NETID;
    Port        := fbWRITEIND.PORT;
    InvokeId        := fbWRITEIND.INVOKEID;
    idxGrp      := fbWRITEIND.IDXGRP;
    idxOffs         := fbWRITEIND.IDXOFFS;
    cbLength        := fbWRITEIND.LENGTH;
    ErrorNumber     := 0;


    CASE idxGrp OF
        16#80000005:

            CASE idxOffs OF
                16#00000007:
                    IF  cbLength <= SIZEOF( arrInt ) THEN
                        IF ( MEMCPY( ADR( arrInt ), fbWRITEIND.DATAADDR, UDINT_TO_INT(cbLength) ) =
 0 ) THEN
                            ErrorNumber := 4000; (*MEMCPY fail*)
                        END_IF
                    ELSE
                        ErrorNumber := 1798;  (* ADS error: invalid parameter value(s)*)
                    END_IF
            ELSE
                ErrorNumber := 1795;    (*ADS error: Invalid index offset*)
            END_CASE

    ELSE
        ErrorNumber := 1794; (*ADS error: invalid index group*)
    END_CASE


    fbWRITEIND(  CLEAR := TRUE );       (*clear indication entry*)


    fbWRITERESBC.NETID := szNetId;
    fbWRITERESBC.PORT := Port;
    fbWRITERESBC.INVOKEID := InvokeId;
    fbWRITERESBC.RESULT := ErrorNumber;
    fbWRITERESBC( RESPOND :=  TRUE );       (*send response*)

END_IF

(*reset fb's*)
IF NOT fbWRITERESBC.BUSY THEN
    fbWRITEIND( CLEAR := FALSE );
    fbWRITERESBC( RESPOND := FALSE );
END_IF
```

Hier können Sie die kompletten Sourcen zu der Beispielapplikation entpacken: https://infosys.beckhoff.com/content/1031/tcplclibadsbc/Resources/12261726475.exe

4 Datentypen

4.1 T_AmsNetId

```
TYPE T_AmsNetId : STRING(23);
END_TYPE
```

Eine SPS-Variable von diesem Typ ist ein String, der die AMS-Netzwerkkennung des Zielgerätes enthält, an das der ADS-Befehl gerichtet wird. Der String besteht aus sechs, durch Punkte getrennten, Zahlenfeldern. Jedes Zahlenfeld enthält eine Zahl zwischen 0 und 254. Gültige AMS-Netzwerkadressen sind z.B. "1.1.1.2.7.1" oder "200.5.7.170.1.7".

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.7.0 Build > 517 TwinCAT v2.8.0 Build > 729	BC9xxx (165) firmware version >= 0xB7	TcAdsBC.Lb6

4.2 T_AmsPort

```
TYPE T_AmsPort : UINT;
END_TYPE
```

ADS-Geräte im TwinCAT-Netzverbund werden durch eine AMS-Netzwerkadresse und eine Portnummer identifiziert. Die Portnummer wird, zusammen mit der Netzwerkadresse, beim Aufruf der ADS-Bausteine als Eingangsparameter benötigt.

Tabelle mit einigen festgelegten ADS-Portnummern:

ADS-Gerät	Portnummer
SPS Laufzeitsystem der Buscontroller BCxxxx	**800**
Nockenschaltwerk	900
SPS Laufzeitsystem 1 auf dem PC SPS Laufzeitsystem 2 auf dem PC SPS Laufzeitsystem 3 auf dem PC SPS Laufzeitsystem 4 auf dem PC	801 811 821 831
NC	500
Reserviert	400
I/O	300
Echtzeitkern	200
Meldesystem (Logger)	100

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.7.0 Build > 517 TwinCAT v2.8.0 Build > 729	BC9xxx (165) firmware version >= 0xB7	TcAdsBC.Lb6

5 Anhang

5.1 Gerätespezifische Fehlercodes: BC9xxx return codes

Wert (hex)	Beschreibung
0x8000	Invalid AMS NetId
0x8001	reserved
0x8002	Tx connection error
0x8003	Connection close error
0x8004	Timeout error
0x8005	Connection is busy
0x8006	Tx error
0x8007	Invalid parameter
0x8010	IP-Connection is in use by another AMS/ADS service
0x8011	Warning, connection already closed
0x8800	No valid network configuration
0x8F00	Resource error
0x8901	ADS length error

Mehr Informationen:
www.beckhoff.de/tx1200

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Deutschland
Telefon: +49 5246 9630
info@beckhoff.de
www.beckhoff.de