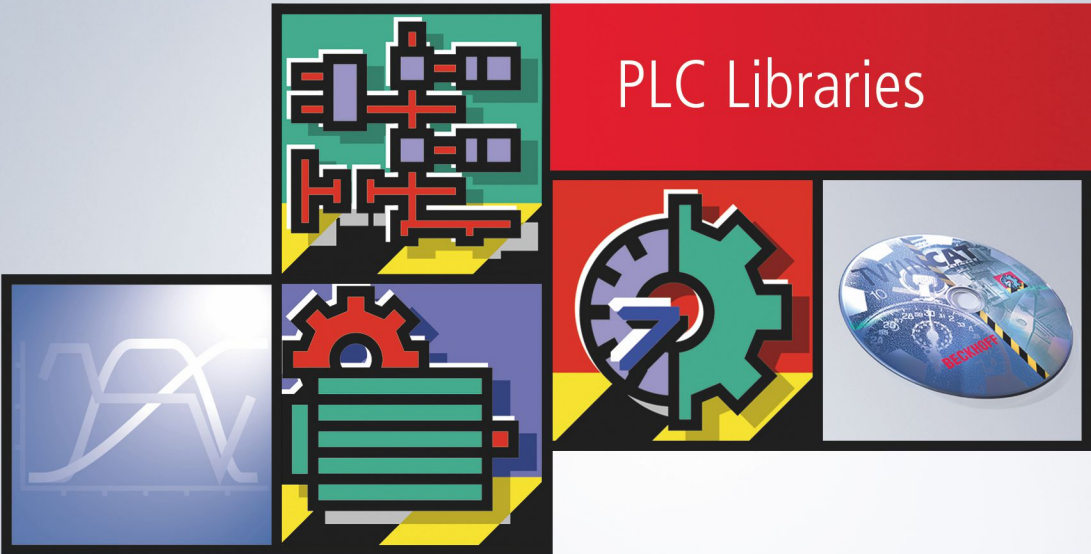


Manual | EN

# TX1200

TwinCAT 2 | PLC Library: TcAdsBC





# Table of contents

<b>1 Foreword</b> .....	<b>5</b>
1.1 Notes on the documentation .....	5
1.2 Safety instructions .....	6
1.3 Notes on information security.....	7
<b>2 Overview</b> .....	<b>8</b>
<b>3 Function blocks</b> .....	<b>9</b>
3.1 ADSREADEX .....	9
3.2 ADSRDWRTEX.....	11
3.3 ADSWRITE .....	13
3.4 ADSCLOSE.....	15
3.5 ADS Indication/Response .....	16
3.5.1 ADSREADIND.....	17
3.5.2 ADSWRITEIND .....	18
3.5.3 ADSRDWRTIND .....	20
3.5.4 ADSREADRESBC .....	21
3.5.5 ADSWRITERESBC.....	22
3.5.6 ADSRDWRTRESBC .....	23
3.5.7 Example 1: ADSREAD Indication/Response .....	24
3.5.8 Example 2: ADSWRITE Indication/Response.....	27
<b>4 Data types</b> .....	<b>31</b>
4.1 T_AmsNetId .....	31
4.2 T_AmsPort .....	31
<b>5 Appendix</b> .....	<b>33</b>
5.1 Device specific error codes: BC9xxx return codes.....	33



# 1 Foreword

## 1.1 Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with applicable national standards.

It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components.

It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

The responsible staff must ensure that the application or use of the products described satisfy all the requirements for safety, including all the relevant laws, regulations, guidelines and standards.

### Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.

We reserve the right to revise and change the documentation at any time and without prior announcement. No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

### Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.

Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

### Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:

EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702  
with corresponding applications or registrations in various other countries.



EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany

### Copyright

© Beckhoff Automation GmbH & Co. KG, Germany.

The reproduction, distribution and utilization of this document as well as the communication of its contents to others without express authorization are prohibited.

Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.

## 1.2 Safety instructions

### Safety regulations

Please note the following safety instructions and explanations!  
Product-specific safety instructions can be found on following pages or in the areas mounting, wiring, commissioning etc.

### Exclusion of liability

All the components are supplied in particular hardware and software configurations appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

### Personnel qualification

This description is only intended for trained specialists in control, automation and drive engineering who are familiar with the applicable national standards.

### Description of symbols

In this documentation the following symbols are used with an accompanying safety instruction or note. The safety instructions must be read carefully and followed without fail!

#### **DANGER**

##### **Serious risk of injury!**

Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.

#### **WARNING**

##### **Risk of injury!**

Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.

#### **CAUTION**

##### **Personal injuries!**

Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.

#### **NOTE**

##### **Damage to the environment or devices**

Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.



##### **Tip or pointer**

This symbol indicates information that contributes to better understanding.

## 1.3 Notes on information security

The products of Beckhoff Automation GmbH & Co. KG (Beckhoff), insofar as they can be accessed online, are equipped with security functions that support the secure operation of plants, systems, machines and networks. Despite the security functions, the creation, implementation and constant updating of a holistic security concept for the operation are necessary to protect the respective plant, system, machine and networks against cyber threats. The products sold by Beckhoff are only part of the overall security concept. The customer is responsible for preventing unauthorized access by third parties to its equipment, systems, machines and networks. The latter should be connected to the corporate network or the Internet only if appropriate protective measures have been set up.

In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our <https://www.beckhoff.com/secguide>.

Beckhoff products and solutions undergo continuous further development. This also applies to security functions. In light of this continuous further development, Beckhoff expressly recommends that the products are kept up to date at all times and that updates are installed for the products once they have been made available. Using outdated or unsupported product versions can increase the risk of cyber threats.

To stay informed about information security for Beckhoff products, subscribe to the RSS feed at <https://www.beckhoff.com/secinfo>.

## 2 Overview

The library contains function blocks for acyclic client-server communication between a **BC9xxx** Bus Controller and other ADS devices in the network. The way in which these function blocks operate is not significantly different from that for the PLC runtime system on the PC.

### Requirements

The upper 4 digits of the AMS NetID (network address) and the TCP/IP address of the ADS target device must be in agreement if the IP connection is to be established.

Example:

ADS communication is to be established with an FC310x Profibus card. The card is assigned its own network address during configuration by the TwinCAT System Manager, e.g.: **'172.16.2.209.4.1'**. This means that it can be considered as an independent ADS device (a remote PC). In order that the IP connection can be established, the first four figures of the network address must agree with the TCP/IP address of the PC in which this card has been configured. This means that the TCP/IP address of the PC must be **'172.16.2.209'**.

### Comments

- The number of connections that may be open simultaneously is restricted to four, in order to maintain the use of resources at a minimum.
- A connection is established automatically when an ADS command is sent.
- A connection is automatically closed after approx. 10 seconds if it has not been used during this time. An unneeded connection can be explicitly closed with the ADSCLOSE function block before this time has elapsed.
- The maximum data size that a BC9xxx Bus Controller can send ( e.g. with ADSWRITE or ADSREADRESBC ) and receive ( e.g. with ADSREAD or ADSWRITEIND ) is limited to 1980 bytes.

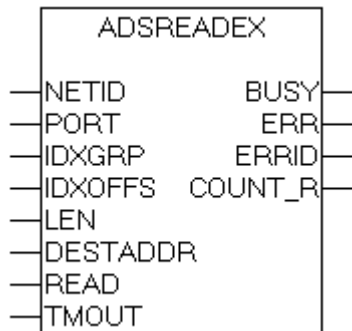
### ADS function blocks

Name	Description
<a href="#">ADSREADEX [► 9]</a>	Read data from an ADS device
<a href="#">ADSRDWRTX [► 11]</a>	Write and read data to/from an ADS device
<a href="#">ADSWRITE [► 13]</a>	Write data to an ADS device
<a href="#">ADSCLOSE [► 15]</a>	Explicitly close the IP connection to another ADS device
<a href="#">ADSREADIND [► 17]</a>	ADSREAD Indication.
<a href="#">ADSWRITEIND [► 18]</a>	ADSWRITE Indication.
<a href="#">ADSRDWRTIND [► 20]</a>	ADSRDWRT Indication.
<a href="#">ADSREADRESBC [► 21]</a>	ADSREAD Response.
<a href="#">ADSWRITERESBC [► 22]</a>	ADSWRITE Response.
<a href="#">ADSRDWRTRESBC [► 23]</a>	ADSRDWRT Response.



### 3 Function blocks

#### 3.1 ADSREADEX



This function block allows execution of an ADS read command, to request data from an ADS device.

##### 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** : is a string containing the AMS network ID of the target device to which the ADS command is directed.

**PORT** : contains the port number of the ADS device.

**IDXGRP** : contains the index group number (32-bit, unsigned) of the requested ADS service. This value is to be found in the ADS table of the addressed device.

**IDXOFFS** : contains the index offset number (32-bit, unsigned) of the requested ADS service. This value is to be found in the ADS table of the addressed device.

**LEN** : contains the number of data to be read in bytes.

**DESTADDR** : contains the address of the buffer which is to receive the data that has been read. The programmer is himself responsible for dimensioning the buffer to a size that can accept 'LEN' bytes. The buffer can be a single variable, an array or a structure, whose address can be found with the ADR operator.

**READ** : the ADS command is triggered by a rising edge at this input.

**TMOUT** : specifies the time until the abortion of the function.

##### VAR\_OUTPUT

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

**BUSY** : if the function block is activated, this output is set. It remains set until feedback is received.

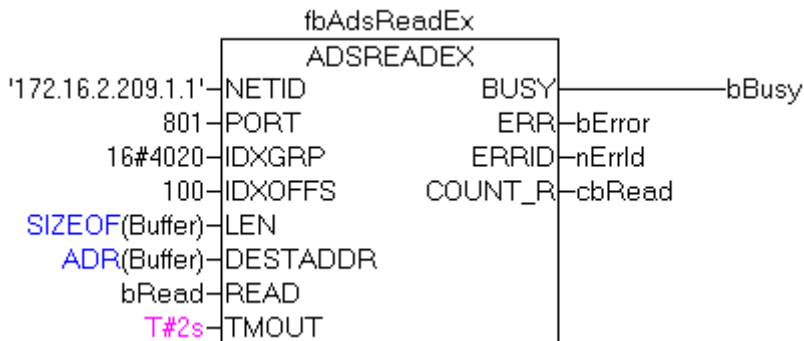
**ERR** : if an ADS error should occur during the transfer of the command, then this output is set once the BUSY output is reset.

**ERRID**: supplies the ADS error number or the device-specific error number [► 33] when the ERR output is set.

**COUNT\_R**: number of successfully read data bytes.

Sample of a call in FBD:

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

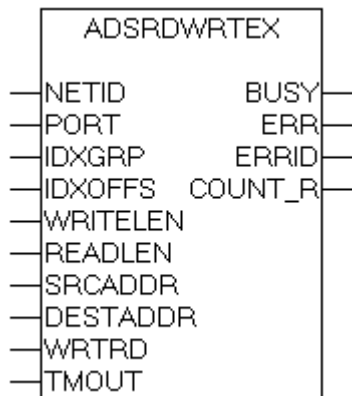


In this sample, an ADS read command is sent to an ADS device with network address '172.16.2.209.1.1' and port number 801. This port number could, for instance, be used to address the PLC's first runtime system. The service to be executed is encoded in the index group and the index offset parameters. Here, 10 bytes of PLC variable data in the flags area are to be read starting from byte offset 100. If successful, 10 bytes of data are copied to the address of the *Buffer* variable.

## Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v2.7.0 & TwinCAT v2.8.0	BC9xxx (165) firmware version >= 0xB6	TcAdsBC.Lb6

## 3.2 ADSRDWRTEX



This function block allows execution of a combined ADS write/read instruction. Data is transmitted to an ADS device (write) and its response data read with one call.

### 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** : is a string containing the AMS network ID of the target device to which the ADS command is directed.

**PORT** : contains the port number of the ADS device.

**IDXGRP** : contains the index group number (32-bit, unsigned) of the requested ADS service. This value is to be found in the ADS table of the addressed device.

**IDXOFFS** : contains the index offset number (32-bit, unsigned) of the requested ADS service. This value is to be found in the ADS table of the addressed device.

**WRITELEN** : contains the number of data to be written in bytes.

**READLEN** : contains the number of data to be read in bytes.

**SRCADDR** : contains the address of the buffer from which the data to be written is to be fetched. The programmer is responsible for dimensioning the buffer such that it can accommodate WRITELEN bytes. The buffer can be a single variable, an array or a structure, whose address can be found with the ADR operator.

**DESTADDR** : contains the address of the buffer which is to receive the data that has been read. The programmer is responsible for dimensioning the buffer such that it can accommodate READLEN bytes. The buffer can be a single variable, an array or a structure, whose address can be found with the ADR operator.

**WRTRD** : the ADS command is triggered by a rising edge at this input.

**TMOUT** : specifies the time until the abortion of the function.

**VAR\_OUTPUT**

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

**BUSY** : if the function block is activated, this output is set. It remains set until feedback is received.

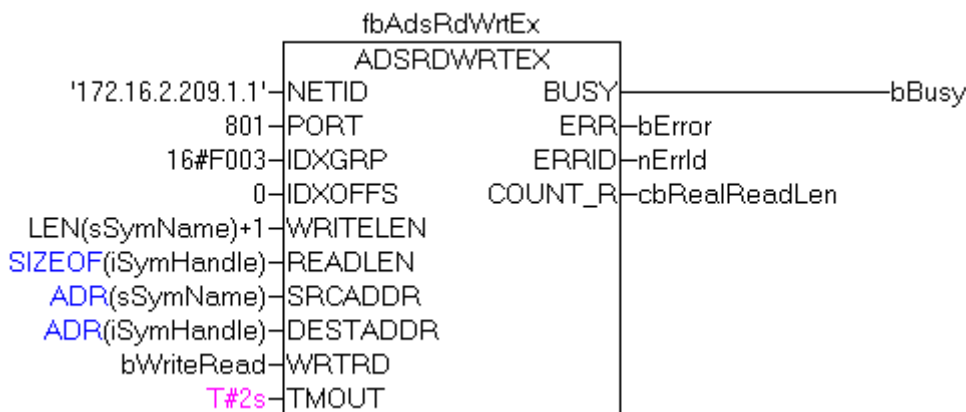
**ERR** : if an ADS error should occur during the transfer of the command, then this output is set once the BUSY output is reset.

**ERRID**: supplies the ADS error number or the device-specific error number [▶ 33] when the ERR output is set.

**COUNT\_R**: number of successfully read data bytes.

Sample of a call in FBD:

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

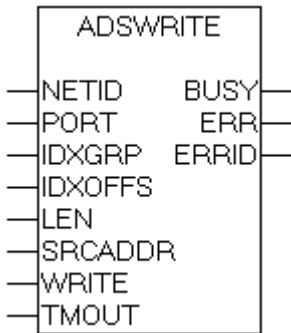


In this sample, an ADS command is sent to an ADS device with network address '172.16.2.209.1.1' and port number 801. Port number 801 could, for instance, be used to address the PLC's first runtime system. The ADS service is encoded in the index group and the index offset. The handle of a PLC variable with the symbol name 'MAIN.VARCOUNTER' is to be read here and returned to the caller. The WRITELEN input parameter contains the string length of the symbol name, plus one byte for the closing NULL. If successful, 4 bytes of data are copied to the address of the *iSymHandle* variable.

**Requirements**

Development environment	Target platform	PLC libraries to include
TwinCAT v2.7.0 & TwinCAT v2.8.0	BC9xxx (165) firmware version >= 0xB6	TcAdsBC.Lb6

### 3.3 ADSWRITE



This function block permits execution of an ADS write command, for the transfer of data to an ADS device.

#### 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** : is a string containing the AMS network ID of the target device to which the ADS command is directed.

**PORT**: contains the port number of the ADS device to which the command is directed.

**IDXGRP** : contains the index group number (32-bit, unsigned) of the requested ADS service. This value is to be found in the ADS table of the addressed device.

**IDXOFFS** : contains the index offset number (32-bit, unsigned) of the requested ADS service. This value is to be found in the ADS table of the addressed device.

**LEN** : contains the number of data to be read in bytes.

**SRCADDR** : contains the address of the buffer from which the data to be written is to be fetched. The programmer is himself responsible for dimensioning the buffer to such a size that 'LEN' bytes can be taken from it. The buffer can be a single variable, an array or a structure, whose address can be found with the ADR operator.

**WRITE** : the ADS command is triggered by a rising edge at this input.

**TMOUT** : specifies the time until the abortion of the function.

**VAR\_OUTPUT**

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

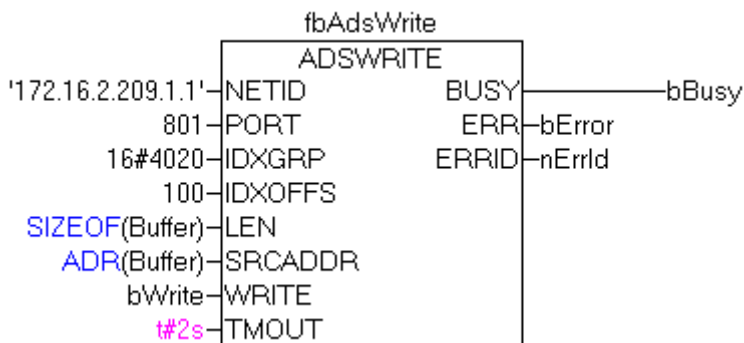
**BUSY** : if the function block is activated, this output is set. It remains set until feedback is received.

**ERR** : if an ADS error should occur during the transfer of the command, then this output is set once the BUSY output is reset.

**ERRID**: supplies the ADS error number or the device-specific error number [▶ 33] when the ERR output is set.

Sample of calling the function block 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
```

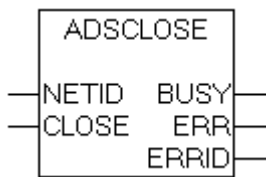


In this sample, an ADS command is sent to an ADS device with network address '172.16.2.209.1.1' and port number 801. This port number could, for instance, be used to address the PLC's first runtime system. The service to be executed is encoded in the index group and the index offset parameters. If successful, 10 bytes of data (*Buffer* variable) are written into the flags area of the target device, starting at byte offset 100.

**Requirements**

Development environment	Target platform	PLC libraries to include
TwinCAT v2.7.0 & TwinCAT v2.8.0	BC9xxx (165) firmware version >= 0xB6	TcAdsBC.Lb6

### 3.4 ADSCLOSE



The ADSCLOSE function block can be used to close explicitly an IP connection that is no longer required. The number of IP connections that may be open simultaneously is restricted to four, in order to maintain the use of resources at a minimum. A connection is automatically established when ADSREADEX, ADSWRITE or ADSRDWRTEX are called. Unused connections are automatically closed after 10 seconds. If it is necessary to establish a connection to more than four ADS devices during that period, the connections that are not required must first be closed.

#### VAR\_INPUT

```

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

[T\\_AmsNetId](#) [► 31]

**NETID**: the AMS network ID of the ADS device whose connection is to be disconnected.

**CLOSE** : the ADS command is triggered by a rising edge at this input.

**TMOUT** : states the time that may not be exceeded by execution of the ADS command.

#### VAR\_OUTPUT

```

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

**BUSY** : if the function block is activated, this output is set. It remains set until feedback is received.

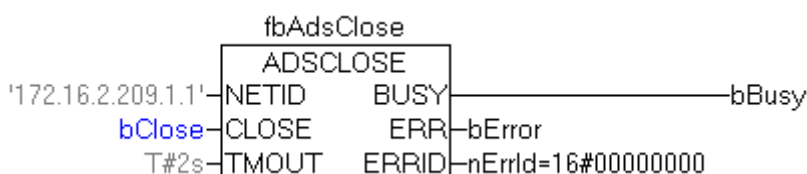
**ERR** : if an ADS error should occur during the transfer of the command, then this output is set once the BUSY output is reset.

**ERRID**: supplies the ADS error number or the [device-specific error number](#) [► 33] when the ERR output is set.

Sample of calling the function block in FBD:

```

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



In this sample, a rising edge at the *bClose* input triggers closure of the IP connection to the ADS device with network address '172.16.2.209.1.1'.

## Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v2.7.0 & TwinCAT v2.8	BC9xxx (165) firmware version >= 0xB6	TcAdsBC.Lb6

## 3.5 ADS Indication/Response

The ADS Indication/Response function blocks enable the establishment of a client-server communication between an ADS device and a PLC task of a bus controller or between two bus controllers (e.g. BC9000). The ADS device can be, for example, a Windows application (uses the AdsDLL/AdsOcx) or another PLC runtime system. Communication between the ADS device and the PLC task is processed using the following service primitives:

- Request
- Indication
- Response
- Confirmation

The communication between an ADS device and a PLC task has the following sequence: an ADS device sends a request to the target device (PLC task). This request is registered in the target device by an indication. The target device (PLC task) thereupon carries out a corresponding service. The service to be carried out is encoded via the index-group/offset parameter. Next the PLC sends a response to the ADS device. The response is registered as confirmation by the ADS source device.

Only one instance of the indication and response function block can meaningfully be used per PLC task. Corresponding with the available ADS services: READ, WRITE and READ & WRITE there is an appropriate indication or response function block for each service.

The ADS devices are addressed via a port address ( PORT ) and a network address (NETID).

### Example:

The PLC task of a BC9000 Bus Controller with the network address "172.64.23.12.1.1" is to be addressed. The PLC task of the Bus Controller has the port number: 800.

The network address:

PORT = 800

NETID = '172.64.23.12.1.1'

### Comments:

- In order for a request to be forwarded to the PLC task, the most significant bit must be set in the IndexGroup parameter during the request, e.g. IG:=0x80000001.
- The maximum data size that a BC9xxx Bus Controller can send ( e.g. with ADSREADRESBC ) and receive ( e.g. with ADSWRITEIND ) is limited to 1980 bytes.

Table 1: ADS Indication/Response function blocks

Service	Name	Description
READ	ADSREADIND [ <a href="#">▶ 17</a> ]	ADSREAD Indication.

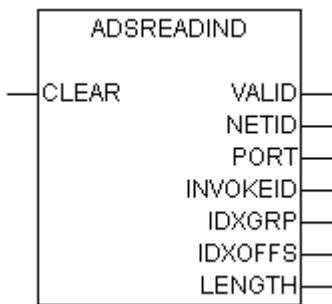


Service	Name	Description
	<a href="#">ADSREADRESBC [▶ 21]</a>	ADSREAD Response
WRITE	<a href="#">ADSWRITEIND [▶ 18]</a>	ADSWRITE Indication
	<a href="#">ADSWRITERESBC [▶ 22]</a>	ADSWRITE Response
READ & WRITE	<a href="#">ADSRDWRRTIND [▶ 20]</a>	ADS-READ & WRITE Indication
	<a href="#">ADSRDWRRTRESBC [▶ 23]</a>	ADS-READ & WRITE Response

**Requirements**

Development environment	Target platform	PLC libraries to include
TwinCAT v2.7.0 Build > 517 TwinCAT v2.8.0 Build > 729	BC9xxx (165) firmware version >= 0xB7	TcAdsBC.Lb6

**3.5.1 ADSREADIND**



The function block registers ADSREAD-Requests at a PLC task as indications and allows them to be processed. The queuing of an indication is reported at the VALID output port by means of a rising edge. The indication is reported as processed via a positive edge at the CLEAR input. A negative edge at the CLEAR input releases the function block for processing further indications. After an indication has been processed a response must be sent to the source device via the [ADSREADRESBC \[▶ 21\]](#) function block. The PORT and NETID parameters can be used to address the source device for this purpose. The INVOKEID parameter is used by the source device to assign the responses to the requests and is also sent back to the source device as a parameter.

**VAR\_INPUT**

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

**CLEAR** : with a rising edge at this input an indication is reported as processed and the outputs of the ADSREADIND function block are reset. A falling edge releases the function block for the processing of further indications.

**VAR\_OUTPUT**

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

**VALID** : the output is set if an indication was registered from the function block and remains set until the latter was reported as processed by a positive edge at the CLEAR input.

**NETID** : is a string containing the AMS network ID of the source device to which the ADS command is directed.

**PORT** : contains the port number of the ADS source device, from which the ADS command was sent.

**INVOKEID** : contains a handle of the command, which was sent. The InvokeId is specified from the source device and serves to identify the commands.

**IDXGRP** : contains the index group number (32-bit, unsigned) of the requested ADS service.

**IDXOFFS** : contains the index offset number (32-bit, unsigned) of the requested ADS service.

**LENGTH** : contains the number of data to be read in bytes.

## Requirements

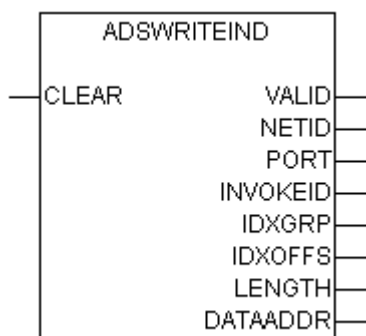
Development environment	Target platform	PLC libraries to include
TwinCAT v2.7.0 Build > 517	BC9xxx (165) firmware version >= 0xB7	TcAdsBC.Lb6
TwinCAT v2.8.0 Build > 729		

## Also see about this

 T\_AmsNetId [[▶ 31](#)]

 T\_AmsPort [[▶ 31](#)]

## 3.5.2 ADSWRITEIND



The function block registers ADSWRITE-Requests to a PLC task as indications and allows them to be processed. The queuing of an indication is reported at the VALID output port by means of a rising edge. The indication is reported as processed via a positive edge at the CLEAR input. A falling edge at the CLEAR input releases the function block for processing further indications. After an indication has been processed a response must be sent to the source device via the [ADSWRITERESBC \[\[▶ 22\]\(#\)\]](#) function block. The PORT and NETID parameters can be used to address the source device for this purpose. The INVOKEID parameter is used by the source device to assign the responses to the requests and is also sent back to the source device as a parameter.

**VAR\_INPUT**

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

**CLEAR** : with a rising edge at this input an indication is reported as processed and the outputs of the ADSWRITEIND function block are reset ( DATAADDR = 0, LENGTH = 0 !). A falling edge releases the function block for the processing of further indications.

**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** : the output is set if an indication was registered from the function block and remains set until the latter was reported as processed by a positive edge at the CLEAR input.

**NETID** : is a string containing the AMS network ID of the source device to which the ADS command is directed.

**PORT** : contains the port number of the ADS source device, from which the ADS command was sent.

**INVOKEID** : contains a handle of the command, which was sent. The InvokeId is specified from the source device and serves to identify the commands.

**IDXGRP** : contains the index group number (32-bit, unsigned) of the requested ADS service.

**IDXOFFS** : contains the index offset number (32-bit, unsigned) of the requested ADS service.



**LENGTH** : contains the length of the written data in bytes.

**DATAADDR** : contains the address of the data buffer, in which the written data is located.

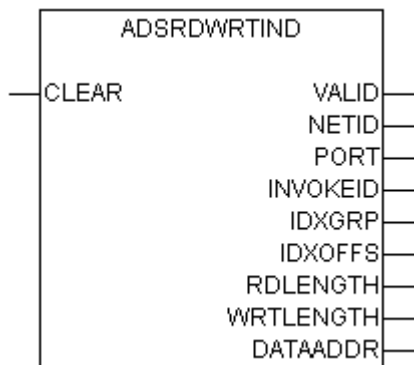
**Requirements**

Development environment	Target platform	PLC libraries to include
TwinCAT v2.7.0 Build > 517 TwinCAT v2.8.0 Build > 729	BC9xxx (165) firmware version >= 0xB7	TcAdsBC.Lb6

**Also see about this**

-  T\_AmsNetId [[▶ 31](#)]
-  T\_AmsPort [[▶ 31](#)]

### 3.5.3 ADSRDWRTIND



The function block registers ADSRDWRT-Requests to a PLC task as indications and allows them to be processed. The queuing of an indication is reported at the VALID output port by means of a rising edge. The indication is reported as processed via a positive edge at the CLEAR input. A falling edge at the CLEAR input releases the function block for processing further indications. After an indication has been processed a response must be sent to the source device via the [ADSRDWRTRESBC \[► 23\]](#) function block. The PORT and NETID parameters can be used to address the source device for this purpose. The INVOKEID parameter is used by the source device to assign the responses to the requests and is also sent back to the source device as a parameter.

#### VAR\_INPUT

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

**CLEAR** : with a rising edge at this input an indication is reported as processed and the outputs of the ADSRDWRTIND function block are reset. A falling edge releases the function block for the processing of further indications.

#### 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** : the output is set if an indication was registered from the function block and remains set until the latter was reported as processed by a positive edge at the CLEAR input.

**NETID** : is a string containing the AMS network ID of the source device to which the ADS command is directed.

**PORT** : contains the port number of the ADS source device, from which the ADS command was sent.

**INVOKEID** : contains a handle of the command, which was sent. The Invokeld is specified from the source device and serves to identify the commands.

**IDXGRP** : contains the index group number (32-bit, unsigned) of the requested ADS service.

**IDXOFFS** : contains the index offset number (32-bit, unsigned) of the requested ADS service.

**LENGTH** : contains the length of data to be read in bytes.

**WRTLENGTH** : contains the length of the written data in bytes.

**DATAADDR** : contains the address of the data buffer, in which the written data is located.

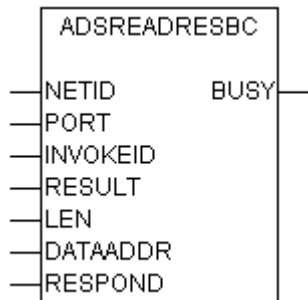
**Requirements**

Development environment	Target platform	PLC libraries to include
TwinCAT v2.7.0 Build > 517 TwinCAT v2.8.0 Build > 729	BC9xxx (165) firmware version >= 0xB7	TcAdsBC.Lb6

**Also see about this**

- 📄 T\_AmsNetId [▶ 31]
- 📄 T\_AmsPort [▶ 31]

**3.5.4 ADSREADRESBC**



The ADSREADRESBC function block is used to acknowledge READ indications of a PLC task. A response is sent to the ADS source device via a positive edge at the RESPOND input. The source device is addressed via the PORT and NETID parameters. The INVOKEID parameter is used by the source device to assign the responses to the requests and is adopted by the output of the [ADSREADIND \[▶ 17\]](#) function block. An error code can be returned to the ADS source device via the RESULT parameter.

**VAR\_INPUT**

```

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

**NETID** : is a string containing the AMS network ID of the source device to which the ADS command is to be sent.

**PORT** : contains the port number of the ADS source device, to which the response should be sent

**INVOKEID** : contains a handle of the command, which was sent. The InvokeId is specified from the source device and serves to identify the commands.

**RESULT** : contains the error code, which should be sent to the source device

**LEN** : contains the number of data to be read in bytes.

**DATAADDR** : contains the address of the data buffer that should be read.

**RESPOND** : the function block is enabled via a positive edge at this input.

#### VAR\_OUTPUT

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

**BUSY** : if the function block is activated, this output is set. It remains set until feedback is received.

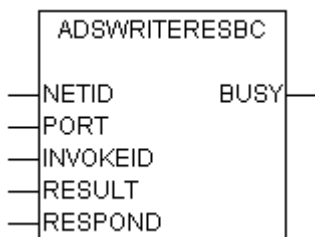
#### Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v2.7.0 Build > 517	BC9xxx (165) firmware version >= 0xB7	TcAdsBC.Lb6
TwinCAT v2.8.0 Build > 729		

#### Also see about this

- 📄 T\_AmsNetId [▶ 31]
- 📄 T\_AmsPort [▶ 31]

### 3.5.5 ADSWRITERESBC



The ADSWRITERESBC function block is used to acknowledge indications of a PLC task. A response is sent to the ADS source device via a positive edge at the RESPOND input. The source device is addressed via the PORT and NETID parameters. The INVOKEID parameter is used by the source device to assign the responses to the requests and is adopted by the output of the [ADSWRITEIND \[▶ 17\]](#) function block. An error code can be returned to the ADS source device via the RESULT parameter.

#### VAR\_INPUT

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

**NETID** : is a string containing the AMS network ID of the source device to which the ADS command is to be sent.

**PORT** : contains the port number of the ADS source device, to which the ADS command is to be sent

**INVOKEID** : contains a handle of the command, which was sent. The InvokeId is specified from the source device and serves to identify the commands.

**RESULT** : contains the error code, which should be sent to the source device

**RESPOND** : the function block is enabled via a positive edge at this input.

**VAR\_OUTPUT**

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

**BUSY** : if the function block is activated, this output is set. It remains set until feedback is received.

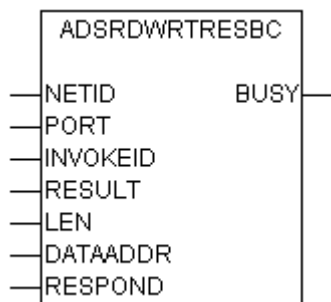
**Requirements**

Development environment	Target platform	PLC libraries to include
TwinCAT v2.7.0 Build > 517 TwinCAT v2.8.0 Build > 729	BC9xxx (165) firmware version >= 0xB7	TcAdsBC.Lb6

**Also see about this**

- 📄 T\_AmsNetId [▶ 31]
- 📄 T\_AmsPort [▶ 31]

**3.5.6 ADSDWRTRESBC**



The ADSDWRTRESBC function block is used to acknowledge indications of a PLC task. A response is sent to the ADS source device via a positive edge at the RESPOND input. The source device is addressed via the PORT and NETID parameters. The INVOKEID parameter is used by the source device to assign the responses to the requests and is adopted by the output of the [ADSDWRTIND \[▶ 20\]](#) function block. An error code can be returned to the ADS source device via the RESULT parameter.

**VAR\_INPUT**

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

```

DATAADDR      : DWORD;
RESPOND       : BOOL;
END_VAR

```

**NETID** : is a string containing the AMS network ID of the source device to which the ADS command is to be sent.

**PORT** : contains the port number of the ADS source device, to which the ADS command is to be sent.

**INVOKEID** : contains a handle of the command, which was sent. The InvokeId is specified from the source device and serves to identify the commands.

**RESULT** : contains the error code, which should be sent to the source device.

**LEN** : length, in bytes, of the read data. This data is sent back to the source device.

**DATAADDR** : address of the data buffer, in which the read data is located.

**RESPOND** : the function block is enabled via a positive edge at this input.

#### VAR\_OUTPUT

```

VAR_OUTPUT
  BUSY      : BOOL;
END_VAR

```


**BUSY** : if the function block is activated, this output is set. It remains set until feedback is received.

#### Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v2.7.0 Build > 517 TwinCAT v2.8.0 Build > 729	BC9xxx (165) firmware version >= 0xB7	TcAdsBC.Lb6

#### Also see about this

 [T\\_AmsNetId \[► 31\]](#)

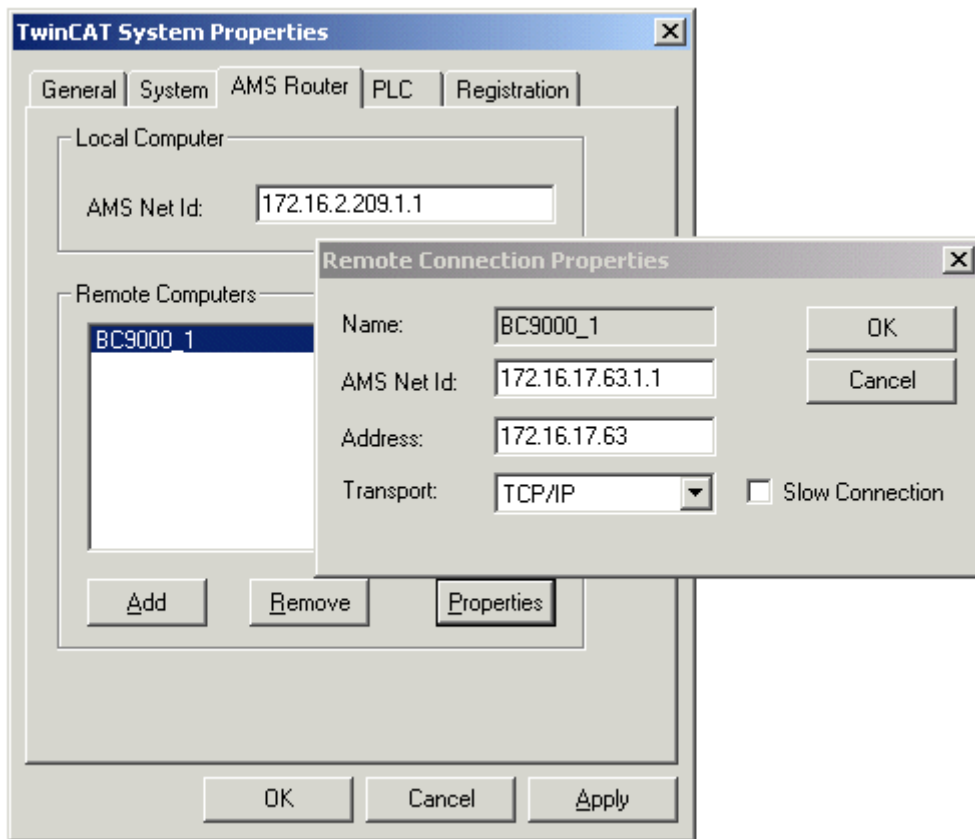
 [T\\_AmsPort \[► 31\]](#)

### 3.5.7 Example 1: ADSREAD Indication/Response

In the sample application, READ requests are sent from a VB application to the PLC task of a BC9000, in order to increment/decrement or reset a PLC counter variable. The READ requests are intercepted as indications at the bus controller, and the desired operation is executed at the counter variable. The bus controller then sends a response with the current value of the counter variable back to the VB application. The value of the counter variable is output on the form. In order to communicate with the PLC task the VB application uses the ActiveX control: AdsOcx. The bus controller must be entered as a remote PC in the TwinCAT system menu in order to be able to route the requests of the VB application to the bus controller via the Ethernet. Here you can unpack the complete <https://infosys.beckhoff.com/content/1033/tcplclibadsbc/Resources/12261725067/.exe>.

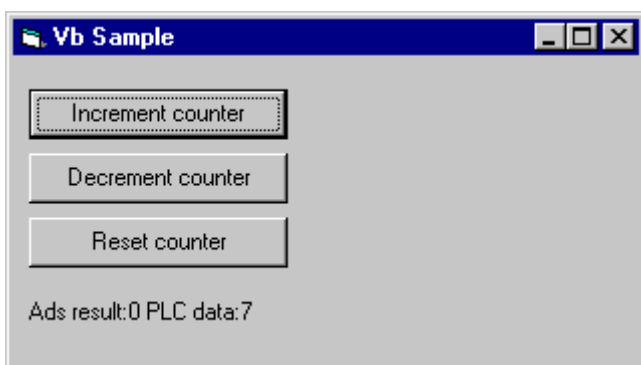


Entering BC9000 as remote PC in the TwinCAT system menu



The TwinCAT system menu can be accessed by clicking the TwinCAT icon->properties in the taskbar. Select *Add* in the *AMS Router* tab. Enter an arbitrary name for the remote PC in the dialog. The network address is entered in *AMS Net Id*. It consists of the IP address of the BC9000 and two further digits: "1.1". The IP address of the BC9000 is entered again in *Address*. The IP address must correspond to the address that was set via the dip switches on the bus controller.

The VB application



A connection to the PLC task of the BC9000 is established in the *Form\_Load* routine. The required service in the PLC task is encoded in the index group parameter:

- IG:0x80000001 -> increment the counter variable;
- IG:0x80000002 -> decrement the counter variable;
- IG:0x80000003 -> set the counter variable = 0;
- IO: 0x0



So that the requests can be routed to the PLC task, the most significant bit must be set in the index group parameter. The values of the index group and index offset parameters can be freely defined. In our example, the bus controller detects whether the PLC variable is to be incremented/decremented or reset via the index group and index offset parameters.

Option Explicit

```
Dim tmpData(1) As Integer
Dim AdsResult As Integer

Private Sub Command1_Click()

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

End Sub

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

Private Sub Command3_Click()
    AdsResult = AdsOcx1.AdsSyncReadReq(&H80000003, &H0, 2, tmpData)
    Labell.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
```

## The PLC program

Create a new PLC project for the bus controller. The following PLC libraries for the bus controller must be integrated under library management: Standard.lb6, PlcHelperBC.lb6 and TcAdsBC.lb6.

The requests are intercepted as indications in the PLC task by an instance of the [ADSREADIND](#) [► 17] function block. Afterwards the index group and index offset parameters and the required data length and validity are checked. In the CASE instruction the desired operation with the PLC variables is carried out. If successful a response is sent back by an instance of the [ADSREADRESBC](#) [► 21] function block to the caller with the current value of the PLC variables. In the case of an error an appropriate error message. At the end the CLEAR and RESPOND flags are reset in order to be able to process further indications.

```
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

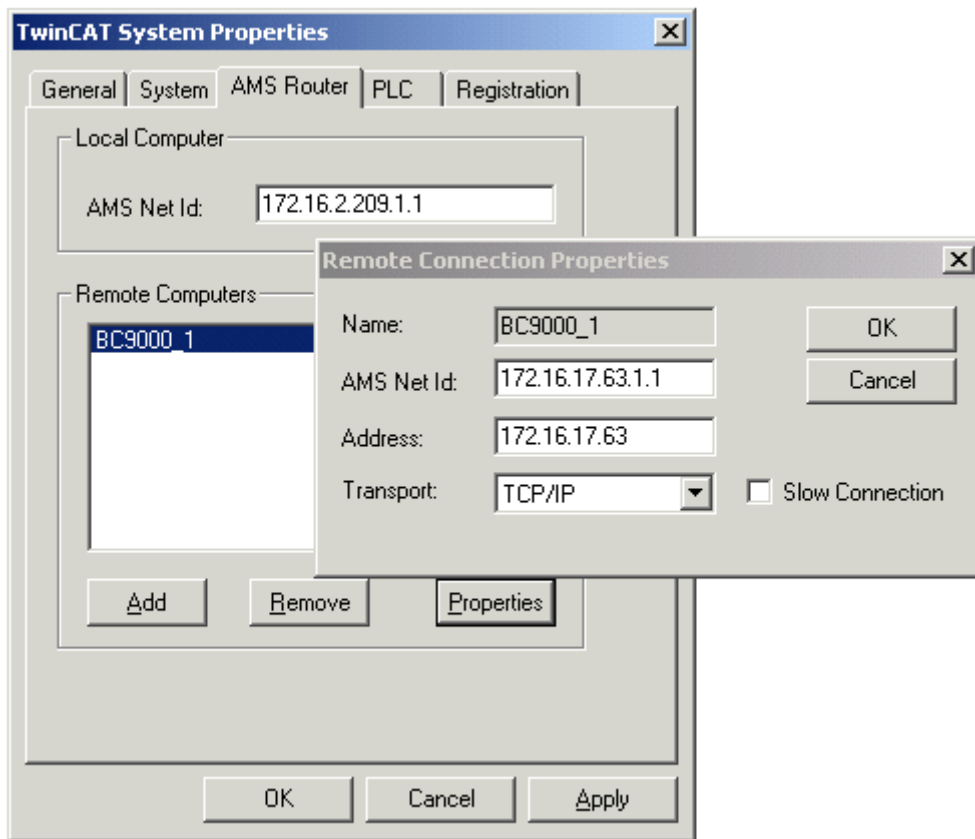
```

Here you can unpack the complete sources for the sample application: <https://infosys.beckhoff.com/content/1033/tcplclibadsbc/Resources/12261725067/.exe>

### 3.5.8 Example 2: ADSWRITE Indication/Response

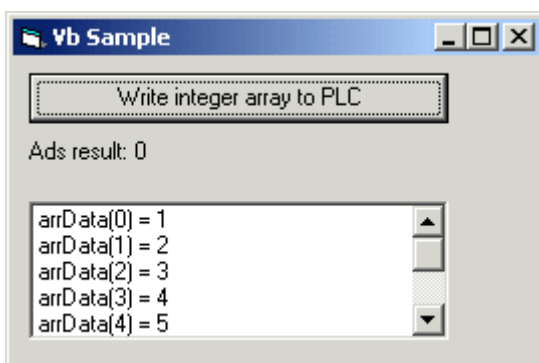
The VB application sends an array with 10 integer values to the PLC task of a BC9000 bus controller. In the PLC task of the bus controller, the data received should be copied into an array variable for further processing. The VB application uses ActiveX Control for communicating with the bus controller: AdsOcx. The bus controller must be entered as a remote PC in the TwinCAT system menu in order to be able to route the requests of the VB application to the bus controller via the Ethernet. Here you can unpack the complete <https://infosys.beckhoff.com/content/1033/tcplclibadsbc/Resources/12261726475/.exe>.

## Entering BC9000 as remote PC in the TwinCAT system menu



The TwinCAT system menu can be accessed by clicking the TwinCAT icon->properties in the taskbar. Select *Add* in the *AMS Router* tab. Enter an arbitrary name for the remote PC in the dialog. The network address is entered in *AMS Net Id*. It consists of the IP address of the BC9000 and two further digits: "1.1". The IP address of the BC9000 is entered again in *Address*. The IP address must correspond to the address that was set via the dip switches on the bus controller.

## The VB application



A connection to the PLC task of the BC9000 Bus Controller is established in the *Form\_Load* routine (port 800 and network address "172.16.17.63.1.1"). The first four digits of the network address correspond to the IP address of the bus controller. The desired service from the PLC task is encoded in the index group and index offset parameters. E.g.:

- IG:0x80000005 and
- IO:0x00000007-> Copy the sent data into the array in the PLC.

The VB application uses the *AdsSyncWriteReq* method to send 20 bytes of data to the bus controller. The values transmitted last are added to a list for monitoring purposes.



So that the requests can be routed to the PLC task, the most significant bit must be set in the index group parameter. The values of the index group and index offset parameters can be freely defined. In our sample, the bus controller detects that it should copy the data received into an array via the index group and index offset parameters.

```
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
```

## The PLC program

Create a new PLC project for the bus controller. The following PLC libraries for the bus controller must be integrated under library management: Standard.lb6, PlcHelperBC.lb6 and TcAdsBC.lb6.

In the PLC task of the bus controller, an instance of the [ADSWRITEIND](#) [► 18] function block is used to receive the data, and an instance of the [ADSWRITERESBC](#) [► 22] function block is used to acknowledge receipt of the data. In the PLC task, the requests are intercepted as so-called indications by the [ADSWRITEIND](#) function block. The validity of the parameters index group and index offset and the transmitted data length is then checked for validity, and the data received are copied into an array variable. A response is then sent back to the caller from an instance of the [ADSWRITERESBC](#) function block (including an error code, if appropriate). This indicates to the VB application that the transmitted data were received successfully.



With the rising edge at the CLEAR input of the [ADSWRITEIND](#) function block the address pointer to the most recently sent data becomes invalid ( = ZERO ). For this reason the sent data is first copied into the PLC variable before the CLEAR input is set to TRUE. At the end the CLEAR and RESPOND flags are reset in order to be able to process further indications.

```
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

```

Here you can unpack the complete sources for the sample application: <https://infosys.beckhoff.com/content/1033/tcplclibadsbc/Resources/12261726475/.exe>

## 4 Data types

### 4.1 T\_AmsNetId

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

A PLC variable of this type is a string containing the AMS network ID of the target device to which the ADS command is directed. The string consists of six numerical fields, separated by dots. Each numerical field contains a number between 0 and 254. Valid AMS network addresses are, for example, "1.1.1.2.7.1" or "200.5.7.170.1.7".

#### Requirements

Development environment	Target platform	PLC libraries to include
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 devices in the TwinCAT network are identified by an AMS network address and a port number. The port number and the network address are both required as input parameters when the ADS blocks are called.

Table with some specified ADS port numbers:

ADS device	Port number
<b>PLC runtime system of the BCxxxx Bus Controllers</b>	<b>800</b>
Cam controller	900
PLC runtime system 1 on the PC	801
PLC runtime system 2 on the PC	811
PLC runtime system 3 on the PC	821
PLC runtime system 4 on the PC	831
NC	500
Reserved	400
I/O	300
Real-time kernel	200
Event System (logger)	100

#### Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v2.7.0 Build > 517 TwinCAT v2.8.0 Build > 729	BC9xxx (165) firmware version >= 0xB7	TcAdsBC.Lb6





## 5 Appendix

### 5.1 Device specific error codes: BC9xxx return codes

Value (hex)	Description
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



More Information:  
[www.beckhoff.com/tx1200](http://www.beckhoff.com/tx1200)

Beckhoff Automation GmbH & Co. KG  
Hülshorstweg 20  
33415 Verl  
Germany  
Phone: +49 5246 9630  
[info@beckhoff.com](mailto:info@beckhoff.com)  
[www.beckhoff.com](http://www.beckhoff.com)

