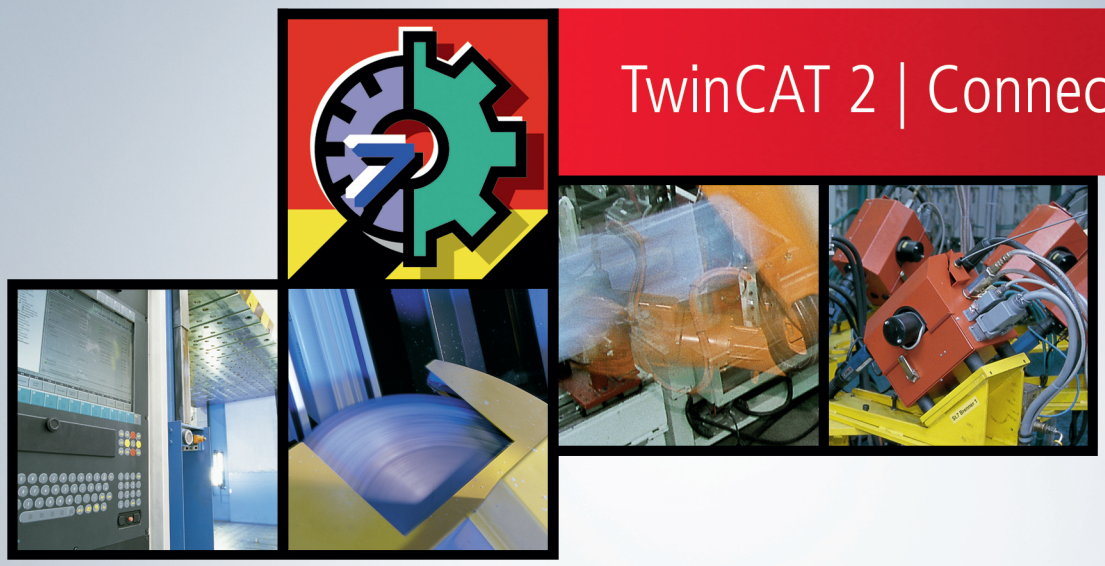
BECKHOFF New Automation Technology

Manual | EN

TX1000

TwinCAT 2 | ADS-Script-DLL



TwinCAT 2 | Connectivity

2022-08-04 | Version: 1.0

Table of contents

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

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 Introduction

The ADS-Script-DLL permits access to ADS devices from script languages (VBScript or JScript). Script languages are used by the Windows Scripting Host (WSH) and in the creation of interactive internet applications. The scripts can be executed either on the web server (active server pages) or at the client (DHTML, ...).

Because the script languages only support a small number of data types, a corresponding method exists in the ADS device for every possible data type. This reduces the number of parameters to a minimum. The methods in which a variable is accessed by name (TcScriptSync::ReadVar() [▶ 12] and TcScriptSync::WriteVar() [▶ 20]) form exceptions to this rule. In these cases the ADS variable data type is found from the ADS device. Both methods convert the parameters internally into the associated data types before making their access to the ADS device.

3 API

Because the script languages only support a small number of data types, a corresponding method exists in the ADS device for every possible data type. This reduces the number of parameters to a minimum. The methods in which a variable is accessed by name (TcScriptSync::ReadVar() [▶ 12] and TcScriptSync::WriteVar() [▶ 20]) form exceptions to this rule. In these cases the ADS variable data type is found from the ADS device. Both methods convert the parameters internally into the associated data types before making their access to the ADS device.

Application		Method	
general	**Communication**	TcScriptSync::ConnectTo() [▶ 10]	
	Read	TcScriptSync::ReadAdsState() [▶ 11]	
	Write	TcScriptSync::WriteAdsState() [▶ 11]	
		by address	**by variable name**
synchronous	**Read**	TcScriptSync::ReadBool() [▶ 13] TcScriptSync::ReadInt8() [▶ 13] TcScriptSync::ReadInt16() [▶ 14] TcScriptSync::ReadInt32() [▶ 14] TcScriptSync::ReadReal32() [▶ 15] TcScriptSync::ReadReal64() [▶ 15] TcScriptSync::ReadArrayOfBool() [▶ 16] TcScriptSync::ReadArrayOfInt8() [▶ 17] TcScriptSync::ReadArrayOfInt16() [▶ 17] TcScriptSync::ReadArrayOfInt32() [▶ 18] TcScriptSync::ReadArrayOfReal32() [▶ 18] TcScriptSync::ReadArrayOfReal64() [▶ 19]	TcScriptSync::ReadVar() [▶ 12]
	Write	TcScriptSync::WriteBool() [▶ 20] TcScriptSync::WriteInt8() [▶ 21] TcScriptSync::WriteInt16() [▶ 21] TcScriptSync::WriteInt32() [▶ 22]	TcScriptSync::WriteVar() [▶ 20]

Application		Method	
		TcScriptSync::WriteReal32() [▶ 22] TcScriptSync::WriteReal64() [▶ 23] TcScriptSync::WriteArrayOfBool() [▶ 24] TcScriptSync::WriteArrayOfInt8() [▶ 24] TcScriptSync::WriteArrayOfInt16() [▶ 25] TcScriptSync::WriteArrayOfInt32() [▶ 25] TcScriptSync::WriteArrayOfReal32() [▶ 26] TcScriptSync::WriteArrayOfReal64() [▶ 27]	

3.1 Method - general

3.1.1 ConnectTo

Opens a communication channel between the object and an ADS device (PLC, NC, ...).

```
object.ConnectTo(
  sAmsNetId As String,
  nPort As Long
)
```

Parameters

sAmsNetId	AdsAmsNetId.
nPort	Port number.

Return value

-

Description

The first parameter contains the six-figure AdsAmsNetId. The individual values are separated from one another by a dot. If an empty string is passed, the local AdsAmsNetId is used.

If a communication channel was already open before the call to *ConnectTo()*, it is automatically closed. The communication channel is also closed if the object is deleted.

Examples

VBScript:

```
Dim TcClientSync
Set TcClientSync =
CreateObject("TCSCRIPT.TcScriptSync")
Call TcClientSync.ConnectTo("", 801)
```

JScript:

```
var TcClientSync;
TcClientSync = new
ActiveXObject("TCSCRIPT.TcScriptSync");
TcClientSync.ConnectTo("172.16.17.13.1.1", 811);
```

3.1.2 ReadAdsState

Reads the current ADS state of an ADS device.

```
object.ReadAdsState() As Long
```

Parameters

-

Return value

ADS-Status.

Description

Any ADS device can adopt several states. The PLC, for instance, can be in either the STOP or the RUN state. This state can be interrogated with the aid of the *ReadAdsState()* method. A summary of all the possible states is found, for instance, in the description of the *ADSSTATE* enum of the ADS-OCX. When evaluating the ADS state, it must be remembered that not every ADS device can adopt every state. The NC, for instance, cannot be in the STOP state.
The most important of the ADS states are RUN (5) and STOP (6), and they are used most often in connection with the PLC.

Examples

VBScript:

```
Dim nAdsState
nAdsState = TcClientSync.ReadAdsState()
WScript.echo "AdsState = ", nAdsState
```

JScript:

```
var nAdsState;
nAdsState = TcClientSync.ReadAdsState();
WScript.echo("AdsState = ", nAdsState);
```

3.1.3 WriteAdsState

Changes the ADS state of an ADS device.

```
object.WriteAdsState(nAdsState As Long)
```

Parameters

nAdsState	new ADS status.

Return value

-

Description

Any ADS device can adopt several states. The PLC, for instance, can be in either the STOP or the RUN state. This state can be changed by means of the *WriteAdsState()* method. A summary of all the possible states is found, for instance, in the description of the *ADSSTATE* enum of the ADS-OCX. When evaluating the ADS state, it must be remembered that not every ADS device can adopt every state. The NC, for instance, cannot be in the STOP state.
The most important of the ADS states are RUN (5) and STOP (6), and they are used most often in connection with the PLC.

Examples

VBScript:

```
Call TcClientSync.WriteAdsState(5)
```

JScript:

```
TcClientSync.WriteAdsState(5);
```

3.2 Method - synchronous

3.2.1 ReadVar

Reads the value of a variable from an ADS device.

```
object.ReadVar(
  sVarName As String
) As Variant
```

Parameters

sVarName	Name of the ADS variable.

Return value

Value of the ADS variable.

Description

The reading procedure is performed synchronously.

Examples

VBScript:

```
Dim VarValue

VarValue = TcClientSync.ReadVar(".PLCVarSInt")

WScript.echo "VarValue = ", VarValue
```

JScript:

```
var VarValue;

VarValue = TcClientSync.ReadVar(".PLCVarSInt");

WScript.echo("VarValue = ", VarValue);
```

3.2.2 ReadBool

Reads a variable of type boolean from an ADS device.

```
object.ReadBool(
  nIndexGroup As Long,
  nIndexOffset As Long
) As Boolean
```

Parameters

nIndexGroup	Index group of the ADS variable.
nIndexOffset	Index offset of the ADS variable.

Return value

Value of the ADS variable.

Description

The reading procedure is performed synchronously. In IEC 61131-6 this corresponds to the BOOL data type.

Examples

VBScript:

```
Dim VarBool
VarBool = TcClientSync.ReadBool(&H4020, 0)
WScript.echo "VarBool = ", VarBool
```

JScript:

```
var VarBool;
VarBool = TcClientSync.ReadBool(0x4020, 0);
WScript.echo("VarBool = ", VarBool);
```

3.2.3 ReadInt8

Reads a variable of type byte from an ADS device.

```
object.ReadInt8(
  nIndexGroup As Long,
  nIndexOffset As Long
) As Byte
```

Parameters

nIndexGroup	Index group of the ADS variable.
nIndexOffset	Index offset of the ADS variable.

Return value

Value of the ADS variable.

Description

The reading procedure is performed synchronously. In IEC 61131-6 this corresponds to the BYTE, SINT or USINT data type.

Examples

VBScript:

```
Dim VarInt8
VarInt8 = TcClientSync.ReadInt8(&H4020, 0)
WScript.echo "VarInt8 = ", VarInt8
```

JScript:

```
var VarInt8;
VarInt8 = TcClientSync.ReadInt8(0x4020, 0);
WScript.echo("VarInt8 = ", VarInt8);
```

3.2.4 ReadInt16

Reads a variable of type integer from an ADS device.

```
object.ReadInt16(
  nIndexGroup  As Long,
  nIndexOffset As Long
) As Integer
```

Parameters

nIndexGroup	Index group of the ADS variable.
nIndexOffset	Index offset of the ADS variable.

Return value

Value of the ADS variable.

Description

The reading procedure is performed synchronously. In IEC 61131-6 this corresponds to the WORD, INT or UINT data type.

Examples

VBScript:

```
Dim VarInt16
VarInt16 = TcClientSync.ReadInt16(&H4020, 0)
WScript.echo "VarInt16 = ", VarInt16
```

JScript:

```
var VarInt16;
VarInt16 = TcClientSync.ReadInt16(0x4020, 0);
WScript.echo("VarInt16 = ", VarInt16);
```

3.2.5 ReadInt32

Reads a variable of type long from an ADS device.

```
object.ReadInt32(
  nIndexGroup As Long,
  nIndexOffset As Long
) As Long
```

Parameters

nIndexGroup	Index group of the ADS variable.
nIndexOffset	Index offset of the ADS variable.

Return value

Value of the ADS variable.

Description

The reading procedure is performed synchronously. In IEC 61131-6 this corresponds to the DWORD, DINT or UDINT data type.

Examples

VBScript:

```
Dim VarInt32
VarInt32 = TcClientSync.ReadInt32(&H4020, 0)
WScript.echo "VarInt32 = ", VarInt32
```

JScript:

```
var VarInt32;
VarInt32 = TcClientSync.ReadInt32(0x4020, 0);
WScript.echo("VarInt32 = ", VarInt32);
```

3.2.6 ReadReal32

Reads a variable of type float from an ADS device.

```
object.ReadReal32(
  nIndexGroup As Long,
  nIndexOffset As Long
) As Single
```

Parameters

nIndexGroup	Index group of the ADS variable.
nIndexOffset	Index offset of the ADS variable.

Return value

Value of the ADS variable.

Description

The reading procedure is performed synchronously. In IEC 61131-6 this corresponds to the REAL data type.

Examples

VBScript:

```
Dim VarReal32
VarReal32 = TcClientSync.ReadReal32(&H4020, 0)
WScript.echo "VarReal32 = ", VarReal32
```

JScript:

```
var VarReal32;
VarReal32 = TcClientSync.ReadReal32(0x4020, 0);
WScript.echo("VarReal32 = ", VarReal32);
```

3.2.7 ReadReal64

Reads a variable of type double from an ADS device.

```
object.ReadReal64(
  nIndexGroup As Long,
  nIndexOffset As Long
) As Double
```

Parameters

nIndexGroup	Index group of the ADS variable.

nIndexOffset	Index offset of the ADS variable.

Return value

Value of the ADS variable.

Description

The reading procedure is performed synchronously. In IEC 61131-6 this corresponds to the LREAL data type.

Examples

VBScript:

```
Dim VarReal64
VarReal64 = TcClientSync.ReadReal64(&H4020, 0)
WScript.echo "VarReal64 = ", VarReal64
```

JScript:

```
var VarReal64;
VarReal64 = TcClientSync.ReadReal64(0x4020, 0);
WScript.echo("VarReal64 = ", VarReal64);
```

3.2.8 ReadArrayOfBool

Reads an array with elements of type boolean from an ADS device.

```
object.ReadArrayOfBool(
  nIndexGroup As Long,
  nIndexOffset As Long,
  nCount As Long
) As Variant
```

Parameters

nIndexGroup	Index group of the ADS variable.
nIndexOffset	Index offset of the ADS variable
nCount	Number of elements to be read.

Return value

The array being read from the ADS device

Description

The read process is executed synchronously. In the IEC 61131-6 the array represents the data type ARRAY OF BOOL..

Samples

VBScript:

```
Dim VarArrayBool
VarArrayBool = TcClientSync.ReadArrayOfBool(&H4020, 0, 2)
WScript.echo "VarArrayBool(0) = ", VarArrayBool(0)
WScript.echo "VarArrayBool(1) = ", VarArrayBool(1)
```

JScript:

```
var VarArrayBool;
VarArrayBool = TcClientSync.ReadArrayOfBool(0x4020, 0, 2);
WScript.echo("VarArrayBool[0] = ",VarArrayBool[0]);
WScript.echo("VarArrayBool[1] = ", VarArrayBool[1]);
```

3.2.9 ReadArrayOfInt8

Reads an array with elements of type byte from an ADS device.

```
object.ReadArrayOfInt8(
  nIndexGroup As Long,
  nIndexOffset As Long,
  nCount As Long
) As Variant
```

Parameters

nIndexGroup	Index group of the ADS variable.
nIndexOffset	Index offset of the ADS variable
nCount	Number of elements to be read.

Return value

The array being read from the ADS device

Description

The read process is executed synchronously. In the IEC 61131-6 the array corresponds to the data type ARRAY OF BYTE, ARRAY OF SINT or ARRAY OF USINT..

Samples

VBScript:

```
Dim VarArrayInt8
VarArrayInt8 = TcClientSync.ReadArrayOfInt8(&H4020, 0, 2)
WScript.echo "VarArrayInt8(0) = ", VarArrayInt8(0)
WScript.echo "VarArrayInt8(1) = ", VarArrayInt8(1)
```

JScript:

```
var VarArrayInt8;
VarArrayInt8 = TcClientSync.ReadArrayOfInt8(0x4020, 0, 2);
WScript.echo("VarArrayInt8[0] = ", VarArrayInt8[0]);
WScript.echo("VarArrayInt8[1] = ", VarArrayInt8[1]);
```

3.2.10 ReadArrayOfInt16

Reads an array with elements of type integer from an ADS device.

```
object.ReadArrayOfInt16(
  nIndexGroup As Long,
  nIndexOffset As Long,
  nCount As Long
) As Variant
```

Parameter

nIndexGroup	Index group of the ADS variable.
nIndexOffset	Index offset of the ADS variable.
nCount	Number of elements which are to be read.

Return value

The array being read from the ADS device

Description

The read process is executed synchronously. In the IEC 61131-6 the array corresponds to the data type ARRAY OF WORD, ARRAY OF INT or ARRAY OF UINT.

Samples

VBScript:

```
Dim VarArrayInt16
VarArrayInt16 = TcClientSync.ReadArrayOfInt16(&H4020, 0,2)
WScript.echo "VarArrayInt16(0) = ", VarArrayInt16(0)
WScript.echo "VarArrayInt16(1) = ", VarArrayInt16(1)
```

JScript:

```
var VarArrayInt16;
VarArrayInt16 = TcClientSync.ReadArrayOfInt16(0x4020, 0, 2);
WScript.echo("VarArrayInt16[0] = ",VarArrayInt16[0]);
WScript.echo("VarArrayInt16[1] = ",VarArrayInt16[1]);
```

3.2.11 ReadArrayOfInt32

Reads an array with elements of type long from an ADS device.

```
object.ReadArrayOfInt32(
  nIndexGroup As Long,
  nIndexOffset As Long,
  nCount As Long
) As Variant
```

Parameters

nIndexGroup	Index group of the ADS variable.
nIndexOffset	Index offset of the ADS variable
nCount	Number of elements to be read.

Return value

The array being read from the ADS device

Description

The read process is executed synchronously. In the IEC 61131-6 the array corresponds to the data type ARRAY OF DWORD, ARRAY OF DINT or ARRAY OF UDINT.

Samples

VBScript:

```
Dim VarArrayInt32
VarArrayInt32 = TcClientSync.ReadArrayOfInt32(&H4020, 0,2)
WScript.echo "VarArrayInt32(0) = ",VarArrayInt32(0)
WScript.echo "VarArrayInt32(1) = ", VarArrayInt32(1)
```

JScript:

```
var VarArrayInt32;
VarArrayInt32 = TcClientSync.ReadArrayOfInt32(0x4020, 0, 2);
WScript.echo("VarArrayInt32[0] = ",VarArrayInt32[0]);
WScript.echo("VarArrayInt32[1] = ",VarArrayInt32[1]);
```

3.2.12 ReadArrayOfReal32

Reads an array with elements of type float from an ADS device.

```
object.ReadArrayOfReal32(
  nIndexGroup As Long,
  nIndexOffset As Long,
  nCount As Long
) As Variant
```

Parameters

nIndexGroup	Index group of the ADS variable.
nIndexOffset	Index offset of the ADS variable
nCount	Number of elements to be read.

Return value

The array being read from the ADS device

Description

The read process is executed synchronously. In the IEC 61131-6 the array corresponds to the data type ARRAY OF REAL.

Samples

VBScript:

```
Dim VarArrayReal32
VarArrayReal32 = TcClientSync.ReadArrayOfReal32(&H4020, 0,2)
WScript.echo "VarArrayReal32(0) = ",VarArrayReal32(0)
WScript.echo "VarArrayReal32(1) = ",VarArrayReal32(1)
```

JScript:

```
var VarArrayReal32;
VarArrayReal32 = TcClientSync.ReadArrayOfReal32(0x4020, 0,2);
WScript.echo("VarArrayReal32[0] = ",VarArrayReal32[0]);
WScript.echo("VarArrayReal32[1] = ",VarArrayReal32[1]);
```

3.2.13 ReadArrayOfReal64

Reads an array with elements of type double from an ADS device.

```
object.ReadArrayOfReal64(
  nIndexGroup As Long,
  nIndexOffset As Long,
  nCount As Long
) As Variant
```

Parameters

nIndexGroup	Index group of the ADS variable.
nIndexOffset	Index offset of the ADS variable
nCount	Number of elements to be read.

Return value

The array being read from the ADS device

Description

The read process is executed synchronously. In the IEC 61131-6 the array corresponds to the data type ARRAY OF LREAL.

Samples

VBScript:

```
Dim VarArrayReal64
VarArrayReal64 = TcClientSync.ReadArrayOfReal64(&H4020, 0,2)
WScript.echo "VarArrayReal64(0) = ",VarArrayReal64(0)
WScript.echo "VarArrayReal64(1) = ",VarArrayReal64(1)
```

JScript:

```
var VarArrayReal64;
VarArrayReal64 = TcClientSync.ReadArrayOfReal64(0x4020, 0,2);
WScript.echo("VarArrayReal64[0] = ",VarArrayReal64[0]);
WScript.echo("VarArrayReal64[1] = ",VarArrayReal64[1]);
```

3.2.14 WriteVar

Writes into a variable in an ADS device.

```
object.WriteVar(
  sVarName As String,
  nValue As Variant
)
```

Parameters

sVarName	Name of the ADS variable.
nValue	The value that is to be written into the variable.

Return value

-

Description

The writing procedure is performed synchronously.

Examples

VBScript:

```
Dim VarValue = 0
Call TcClientSync.WriteVar(".PLCSIntVar", VarValue)
```

JScript:

```
var VarValue =0;
TcClientSync.WriteVar(".PLCSIntVar",VarValue);
```

3.2.15 WriteBool

Writes a variable of type boolean.

```
object.WriteBool(
  nIndexGroup As Long,
  nIndexOffset As Long,
  bValue As Boolean
)
```

Parameters

nIndexGroup	Index group of the ADS variable.
nIndexOffset	Index offset of the ADS variable.
bValue	The value that is to be written into the variable.

Return value

-

Description

The writing procedure is performed synchronously. In IEC 61131-6 this corresponds to the BOOL data type.

Examples

VBScript:

```
Dim VarBool = 0
Call TcClientSync.WriteBool(&H4020, 0, VarBool)
```

JScript:

```
var VarBool =0;
TcClientSync.WriteBool(0x4020, 0, VarBool);
```

3.2.16 WriteInt8

Writes a variable of type byte.

```
object.WriteInt8(
  nIndexGroup As Long,
  nIndexOffset As Long,
  nValue As Byte
)
```

Parameters

nIndexGroup	Index group of the ADS variable.
nIndexOffset	Index offset of the ADS variable.
nValue	The value that is to be written into the variable.

Return value

-

Description

The writing procedure is performed synchronously. In IEC 61131-6 this corresponds to the BYTE, SINT or USINT data type.

Examples

VBScript:

```
Dim VarInt8 = 0
Call TcClientSync.WriteInt8(&H4020, 0, VarInt8)
```

JScript:

```
var VarInt8 =0;
TcClientSync.WriteInt8(0x4020, 0, VarInt8);
```

3.2.17 WriteInt16

Writes a variable of type integer.

```
object.WriteInt16(
  nIndexGroup As Long,
  nIndexOffset As Long,
  nValue As Integer
)
```

Parameters

nIndexGroup	Index group of the ADS variable.
nIndexOffset	Index offset of the ADS variable.
nValue	The value that is to be written into the variable.

Return value

-

Description

The writing procedure is performed synchronously. In IEC 61131-6 this corresponds to the WORD, INT or UINT data type.

Examples

VBScript:

```
Dim VarInt16 = 0
Call TcClientSync.WriteInt16(&H4020, 0, VarInt16)
```

JScript:

```
var VarInt16 =0;
TcClientSync.WriteInt16(0x4020, 0, VarInt16);
```

3.2.18 WriteInt32

Writes a variable of type long.

```
object.WriteInt32(
  nIndexGroup As Long,
  nIndexOffset As Long,
  nValue As Long
)
```

Parameters

nIndexGroup	Index group of the ADS variable.
nIndexOffset	Index offset of the ADS variable.
nValue	The value that is to be written into the variable.

Return value

-

Description

The writing procedure is performed synchronously. In IEC 61131-6 this corresponds to the DWORD, DINT or UDINT data type.

Examples

VBScript:

```
Dim VarInt32 = 0
Call TcClientSync.WriteInt32(&H4020, 0, VarInt32)
```

JScript:

```
var VarInt32 =0;
TcClientSync.WriteInt32(0x4020, 0, VarInt32);
```

3.2.19 WriteReal32

Writes a variable of type single.

```
object.WriteReal32(
  nIndexGroup As Long,
  nIndexOffset As Long,
  fValue As Single
)
```

Parameters

nIndexGroup	Index group of the ADS variable.
nIndexOffset	Index offset of the ADS variable.
fValue	The value that is to be written into the variable.

Return value

-

Description

The writing procedure is performed synchronously. In IEC 61131-6 this corresponds to the REAL data type.

Examples

VBScript:

```
Dim VarReal32 = 0
Call TcClientSync.WriteReal32(&H4020, 0, VarReal32)
```

JScript:

```
var VarReal32 =0;
TcClientSync.WriteReal32(0x4020, 0, VarReal32);
```

3.2.20 WriteReal64

Writes a variable of type double.

```
object.WriteReal64(
  nIndexGroup As Long,
  nIndexOffset As Long,
  fValue As Double
)
```

Parameters

nIndexGroup	Index group of the ADS variable.
nIndexOffset	Index offset of the ADS variable.
fValue	The value that is to be written into the variable.

Return value

-

Description

The writing procedure is performed synchronously. In IEC 61131-6 this corresponds to the LREAL data type.

Examples

VBScript:

```
Dim VarReal64 = 0
Call TcClientSync.WriteReal64(&H4020, 0, VarReal64)
```

JScript:

```
var VarReal64 =0;
TcClientSync.WriteReal64(0x4020, 0, VarReal64);
```

3.2.21 WriteArrayOfBool

Writes an array with elements of type boolean to an ADS device.

```
object.WriteArrayOfBool(
  nIndexGroup As Long,
  nIndexOffset As Long,
  pValue As Variant
)
```

Parameters

nIndexGroup	Index group of the ADS variable.
nIndexOffset	Index offset of the ADS variable
nCount	Number of elements to write.

Return value

-

Description

The write process is executed synchronously. In the IEC 61131-6 the array corresponds to the data type ARRAY OF BOOL.

Samples

VBScript:

```
Dim VarArrayBool(1)
VarArrayBool(0) = True
VarArrayBool(1) = False
Call TcClientSync.WriteArrayOfBool(&H4020, 0, VarArrayBool)
```

JScript:

```
var VarArrayBool[2];
VarArrayBool[0] = true;
VarArrayBool[1] = false;
TcClientSync.WriteArrayOfBool(0x4020, 0, VarArrayBool);
```

3.2.22 WriteArrayOfInt8

Writes an array with elements of type byte to an ADS device.

```
object.WriteArrayOfInt8(
  nIndexGroup As Long,
  nIndexOffset As Long,
  pValue As Variant
)
```

Parameters

nIndexGroup	Index group of the ADS variable.
nIndexOffset	Index offset of the ADS variable
nCount	Number of elements to write.

Return value

-

Description

The write process is executed synchronously. In the IEC 61131-6 the array corresponds to the data type ARRAY OF BYTE, ARRAY OF SINT oder ARRAY OF USINT.

Samples

VBScript:

```
Dim VarArrayInt8(1)
VarArrayInt8(0) = 0
VarArrayInt8(1) = 1
Call TcClientSync.WriteArrayOfInt8(&H4020, 0, VarArrayInt8)
```

JScript:

```
var VarArrayInt8[2];
VarArrayInt8[0] = 0;
VarArrayInt8[1] = 1;
TcClientSync.WriteArrayOfInt8(0x4020, 0, VarArrayInt8);
```

3.2.23 WriteArrayOfInt16

Writes an array with elements of type integer to an ADS device.

```
object.WriteArrayOfInt16(
  nIndexGroup As Long,
  nIndexOffset As Long,
  pValue As Variant
)
```

Parameters

nIndexGroup	Index group of the ADS variable.
nIndexOffset	Index offset of the ADS variable
nCount	Number of elements to write.

Return value

-

Description

The write process is executed synchronously. In the IEC 61131-6 the array corresponds to the data type ARRAY OF WORD, ARRAY OF INT oder ARRAY OF UINT.

Samples

VBScript:

```
Dim VarArrayInt16(1)
VarArrayInt16(0) = 0
VarArrayInt16(1) = 1
Call TcClientSync.WriteArrayOfInt16(&H4020, 0, VarArrayInt16)
```

JScript:

```
var VarArrayInt16[2];
VarArrayInt16[0] = 0;
VarArrayInt16[1] = 1;
TcClientSync.WriteArrayOfInt16(0x4020, 0,
VarArrayInt16);
```

3.2.24 WriteArrayOfInt32

Writes an array with elements of type long to an ADS device.

```
object.WriteArrayOfInt32(
  nIndexGroup As Long,
  nIndexOffset As Long,
  pValue As Variant
)
```

Parameters

nIndexGroup	Index group of the ADS variable.
nIndexOffset	Index offset of the ADS variable
nCount	Number of elements to write.

Return value

-

Description

The write process is executed synchronously. In the IEC 61131-6 the array corresponds to the data type ARRAY OF DWORD, ARRAY OF DINT or ARRAY OF UDINT.

Samples

VBScript:

```
Dim VarArrayInt32(1)
VarArrayInt32(0) = 0
VarArrayInt32(1) = 1
Call TcClientSync.WriteArrayOfInt32(&H4020, 0, VarArrayInt32)
```

JScript:

```
var VarArrayInt32[2];
VarArrayInt32[0] = 0;
VarArrayInt32[1] = 1;
TcClientSync.WriteArrayOfInt32(0x4020, 0, VarArrayInt32);
```

3.2.25 WriteArrayOfReal32

Writes an array with elements of type single to an ADS device.

```
object.WriteArrayOfReal32(
  nIndexGroup As Long,
  nIndexOffset As Long,
  pValue As Variant
)
```

Parameters

nIndexGroup	Index group of the ADS variable.
nIndexOffset	Index offset of the ADS variable
nCount	Number of elements to write.

Return value

-

Description

The write process is executed synchronously. In the IEC 61131-6 the array corresponds to the data type ARRAY OF REAL.

Samples

VBScript:

```
Dim VarArrayReal32(1)
VarArrayReal32(0) = 0.1
VarArrayReal32(1) = 1.2
Call TcClientSync.WriteArrayOfReal32(&H4020, 0, VarArrayReal32)
```

JScript:

```
var VarArrayReal32[2];
VarArrayReal32[0] = 0.1;
VarArrayReal32[1] = 1.2;
TcClientSync.WriteArrayOfReal32(0x4020, 0, VarArrayReal32);
```

3.2.26 WriteArrayOfReal64

Writes an array with elements of type double to an ADS device.

```
object.WriteArrayOfReal64(
  nIndexGroup As Long,
  nIndexOffset As Long,
  pValue As Variant
)
```

Parameters

nIndexGroup	Index group of the ADS variable.
nIndexOffset	Index offset of the ADS variable
nCount	Number of elements to write.

Return value

-

Description

The write process is executed synchronously. In the IEC 61131-6 the array corresponds to the data type ARRAY OF LREAL.

Samples

VBScript:

```
Dim VarArrayReal64(1)
VarArrayReal64(0) = 0.1
VarArrayReal64(1) = 1.2
Call TcClientSync.WriteArrayOfReal64(&H4020, 0, VarArrayReal64)
```

JScript:

```
var VarArrayReal64[2];
VarArrayReal64[0] = 0.1;
VarArrayReal64[1] = 1.2;
TcClientSync.WriteArrayOfReal64(0x4020, 0, VarArrayReal64);
```

4 Examples

Description	Source text
Example 1: Windows Scripting Host [▶ 28]	https://infosys.beckhoff.com/content/1033/ tcscriptdll/ Resources/12459716107/.zip
Example 2: Visual Basic	https://infosys.beckhoff.com/content/1033/ tcscriptdll/ Resources/12459720971/.zip
Example 3: Active Server Pages [▶ 36]	https://infosys.beckhoff.com/content/1033/ tcscriptdll/ Resources/12459711755/.zip

Also see about this

- Reading and writing of PLC variables [▶ 30]

4.1 Windows Scripting Host

Load the PLC program into the local PLC on your PC and run the PLC program. Then call the script.

After the communication channel has been opened, a check is made as to whether the PLC is in the RUN state. If not, the PLC is started. The next step is that all the PLC variables are read off and individually displayed. Then all the variables are incremented and are checked by reading off and displaying them again. The PLC is finally stopped.

VBScript:

```
Dim TcClientSync
Dim VarBool, VarInt8, VarInt16, VarInt32, VarReal32,VarReal64

'create TcScript-Object
Set TcClientSync =CreateObject("TCSCRIPT.TcScriptSync")
Call TcClientSync.ConnectTo("", 801)

if (TcClientSync.ReadAdsState() <> 5) then
  WScript.echo("PLC is not running!" & vbCrLf& "Start PLC.")
  Call TcClientSync.WriteAdsState(5)
end if

'read from PLC
VarBool = TcClientSync.ReadBool(&H4020, 0)
WScript.echo "VarBool = ", VarBool

VarInt8 = TcClientSync.ReadInt8(&H4020, 2)
WScript.echo "VarInt8 = ", VarInt8

VarInt16 = TcClientSync.ReadInt16(&H4020, 4)
WScript.echo "VarInt16 = ", VarInt16

VarInt32 = TcClientSync.ReadInt32(&H4020, 6)
WScript.echo "VarInt32 = ", VarInt32

VarReal32 = TcClientSync.ReadReal32(&H4020, 10)
WScript.echo "VarReal32 = ", VarReal32

VarReal64 = TcClientSync.ReadReal64(&H4020, 14)
WScript.echo "VarReal64 = ", VarReal64

'write to PLC
Call TcClientSync.WriteBool(&H4020, 0, NOT VarBool)
Call TcClientSync.WriteInt8(&H4020, 2, VarInt8 + 1)
Call TcClientSync.WriteInt16(&H4020, 4, VarInt16 +1)
Call TcClientSync.WriteInt32(&H4020, 6, VarInt32 + 1)
Call TcClientSync.WriteReal32(&H4020, 10, VarReal32 +1.1)
Call TcClientSync.WriteReal64(&H4020, 14, VarReal64 +1.11)

Call TcClientSync.WriteVar(".PLCBoolVar", VarBool)
Call TcClientSync.WriteVar(".PLCSIntVar", VarInt8 +2)
Call TcClientSync.WriteVar(".PLCIntVar", VarInt16 +2)
Call TcClientSync.WriteVar(".PLCDIntVar", VarInt32 +2)
Call TcClientSync.WriteVar(".PLCRealVar", VarReal32 +2.1)
```

```
Call TcClientSync.WriteVar(".PLCLRealVar", VarReal64 +2.11)

'read again from PLC
VarBool = TcClientSync.ReadVar(".PLCBoolVar")
WScript.echo "VarBool = ", VarBool

VarInt8 = TcClientSync.ReadVar(".PLCSIntVar")
WScript.echo "VarInt8 = ", VarInt8

VarInt16 = TcClientSync.ReadVar(".PLCIntVar")
WScript.echo "VarInt16 = ", VarInt16

VarInt32 = TcClientSync.ReadVar(".PLCDIntVar")
WScript.echo "VarInt32 = ", VarInt32

VarReal32 = TcClientSync.ReadVar(".PLCRealVar")
WScript.echo "VarReal32 = ", VarReal32

VarReal64 = TcClientSync.ReadVar(".PLCLRealVar")
WScript.echo "VarReal64 = ", VarReal64

if (TcClientSync.ReadAdsState() = 5) then
  WScript.echo("PLC is running!" & vbCrLf & "Stop PLC.")
  Call TcClientSync.WriteAdsState(6)
end if
```

JScript:

```
var TcClientSync;
var VarBool, VarInt8, VarInt16, VarInt32, VarReal32, VarReal64;

// create TcScript-Object
TcClientSync = new ActiveXObject("TCSCRIPT.TcScriptSync");
TcClientSync.ConnectTo("", 801);

if (TcClientSync.ReadAdsState() != 5)
{
  WScript.echo("PLC is not running!\nStart PLC.");
  TcClientSync.WriteAdsState(5);
}

// read from PLC
VarBool = TcClientSync.ReadBool(0x4020, 0);
WScript.echo("VarBool = ", VarBool);

VarInt8 = TcClientSync.ReadInt8(0x4020, 2);
WScript.echo("VarInt8 = ", VarInt8);

VarInt16 = TcClientSync.ReadInt16(0x4020, 4);
WScript.echo("VarInt16 = ", VarInt16);

VarInt32 = TcClientSync.ReadInt32(0x4020, 6);
WScript.echo("VarInt32 = ", VarInt32);

VarReal32 = TcClientSync.ReadReal32(0x4020, 10);
WScript.echo("VarReal32 = ", VarReal32);

VarReal64 = TcClientSync.ReadReal64(0x4020, 14);
WScript.echo("VarReal64 = ", VarReal64);

// write to PLC
TcClientSync.WriteBool(0x4020, 0, !VarBool);
TcClientSync.WriteInt8(0x4020, 2, VarInt8 + 1);
TcClientSync.WriteInt16(0x4020, 4, VarInt16 + 1);
TcClientSync.WriteInt32(0x4020, 6, VarInt32 + 1);
TcClientSync.WriteReal32(0x4020, 10, VarReal32 + 1.1);
TcClientSync.WriteReal64(0x4020, 14, VarReal64 + 1.11);

TcClientSync.WriteVar(".PLCBoolVar", VarBool);
TcClientSync.WriteVar(".PLCSIntVar", VarInt8 +2);
TcClientSync.WriteVar(".PLCIntVar", VarInt16 + 2);
TcClientSync.WriteVar(".PLCDIntVar", VarInt32 + 2);
TcClientSync.WriteVar(".PLCRealVar", VarReal32 +2.1);
TcClientSync.WriteVar(".PLCLRealVar", VarReal64 +2.11);

// read again from PLC
VarBool = TcClientSync.ReadVar(".PLCBoolVar");
WScript.echo("VarBool = ", VarBool);

VarInt8 = TcClientSync.ReadVar(".PLCSIntVar");
```

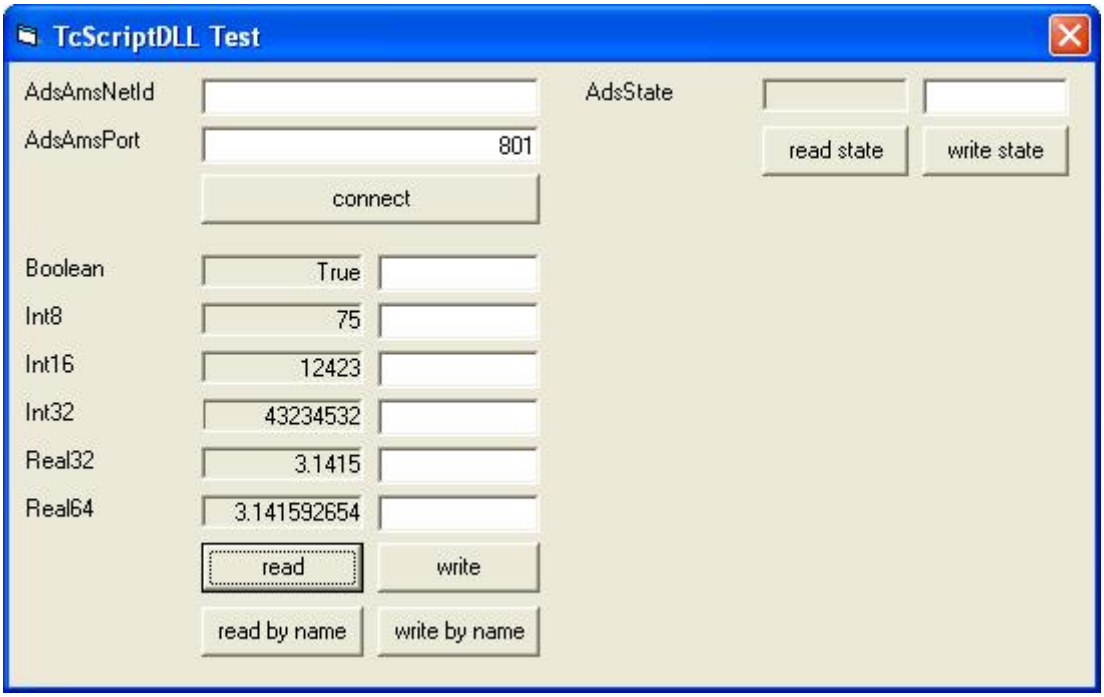
```
WScript.echo("VarInt8 = ", VarInt8);

VarInt16 = TcClientSync.ReadVar(".PLCIntVar");
WScript.echo("VarInt16 = ", VarInt16);

VarInt32 = TcClientSync.ReadVar(".PLCDIntVar");
WScript.echo("VarInt32 = ", VarInt32);

VarReal32 = TcClientSync.ReadVar(".PLCRealVar");
WScript.echo("VarReal32 = ", VarReal32);

VarReal64 = TcClientSync.ReadVar(".PLCLRealVar");
WScript.echo("VarReal64 = ", VarReal64);

if (TcClientSync.ReadAdsState() == 5)
{
  WScript.echo("PLC is running!\nStop PLC.");
  TcClientSync.WriteAdsState(6);
}
```

Unpack the example program https://infosys.beckhoff.com/content/1033/ tcscriptdll/Resources/12459716107/.zip 'Windows Scripting Host'.

4.2 Reading and writing of PLC variables

As mentioned in the introduction, the ADS Script DLL is primarily designed for scripting languages. Nevertheless, at this point is a Visual Basic program. It only serves to better demonstrate the capabilities of the ADS Script DLL. For applications that are to be created with Visual Basic, the ADS-OCX is better suited.

First load the PLC program into the local PLC (runtime system 1) on your PC. Then start the Visual Basic program:



Unless you provide some other indication, a communication channel is created to your local PLC. By entering a different AdsAmsNetId and port number, you can connect to a different runtime system on another PC. The above-mentioned PLC program, however, must still be active on that other runtime system. It is also necessary that a remote computer, if selected, is entered into the TwinCAT System Control.

You can either read the PLC variables by address (IndexGroup and IndexOffset), or by variable name. The same applies to writing new values. This has no meaning for the operator. It is only to list the two possibilities.

In the upper right corner the state of the PLC can be read and changed. A 5 corresponds to the RUN state, while a 6 corresponds to STOP.

Visual Basic 6 program

```
Option Explicit


Public TcClientSync As TCSCRIPTLib.TcScriptSync

Public nAdsAmsPort As Integer

Public strAdsAmsNetId As String


Private Sub Form_Load()

    Set TcClientSync =
CreateObject("TcScript.TcScriptSync")

    Call TcClientSync.ConnectTo("",
801)

End Sub


Private Sub Form_Terminate()

    Set TcClientSync = Nothing

End Sub


Private Sub cmdConnect_Click()

    nAdsAmsPort = CLng(txtAdsAmsPort.Text)

    if (Len(txtAdsAmsNetId.Text) > 0) Then

       strAdsAmsNetId =
txtAdsAmsNetId.Text

    Else

       strAdsAmsNetId =
""

    End If

    Set TcClientSync =
CreateObject("TcScript.TcScriptSync")

    Call TcClientSync.ConnectTo(strAdsAmsNetId,
nAdsAmsPort)

End Sub


Private Sub cmdRead_Click()

    Dim VarBool As Variant

    Dim VarInt8 As Byte

    Dim VarInt16 As Integer

    Dim VarInt32 As Long

    Dim VarReal32 As Single

    Dim VarReal64 As Double


    VarBool = TcClientSync.ReadBool(&H4020,
0)

    lblBoolean.Caption = VarBool
```

```
    VarInt8 = TcClientSync.ReadInt8(&H4020,
2)

    lblInt8.Caption = VarInt8


    VarInt16 = TcClientSync.ReadInt16(&H4020,
4)

    lblInt16.Caption = VarInt16


    VarInt32 = TcClientSync.ReadInt32(&H4020,
6)

    lblInt32.Caption = VarInt32


    VarReal32 = TcClientSync.ReadReal32(&H4020,
10)

    lblReal32.Caption = VarReal32


    VarReal64 = TcClientSync.ReadReal64(&H4020,
14)

    lblReal64.Caption = VarReal64

End Sub


Private Sub cmdReadByName_Click()

    Dim VarBool As Variant

    Dim VarInt8 As Byte

    Dim VarInt16 As Integer

    Dim VarInt32 As Long

    Dim VarReal32 As Single

    Dim VarReal64 As Double


    VarBool =
TcClientSync.ReadBoolVar(".PLCBoolVar")

    lblBoolean.Caption = VarBool


    VarInt8 =
TcClientSync.ReadInt8Var(".PLCSIntVar")

    lblInt8.Caption = VarInt8


    VarInt16 =
TcClientSync.ReadInt16Var(".PLCIntVar")

    lblInt16.Caption = VarInt16


    VarInt32 =
TcClientSync.ReadInt32Var(".PLCDIntVar")

    lblInt32.Caption = VarInt32


    VarReal32 =
```

```
TcClientSync.ReadReal32Var(".PLCRealVar")

    lblReal32.Caption = VarReal32



    VarReal64 =
TcClientSync.ReadReal64Var(".PLCLRealVar")

    lblReal64.Caption = VarReal64

End Sub


Private Sub cmdWrite_Click()

    Dim VarBool As Boolean

    Dim VarInt8 As Byte

    Dim VarInt16 As Integer

    Dim VarInt32 As Long

    Dim VarReal32 As Single

    Dim VarReal64 As Double



    If (txtBoolean.Text <> "")
Then

        VarBool =
CBool(txtBoolean.Text)

        Call
TcClientSync.WriteBool(&H4020, 0, VarBool)

        txtBoolean.Text =
VarBool

    End If



    If (txtInt8.Text <> "")
Then

        VarInt8 =
CByte(txtInt8.Text)

        Call
TcClientSync.WriteInt8(&H4020, 2, VarInt8)

        txtInt8.Text =
VarInt8

    End If



    If (txtInt16.Text <> "")
Then

        VarInt16 =
CInt(txtInt16.Text)

        Call
TcClientSync.WriteInt16(&H4020, 4, VarInt16)

        txtInt16.Text =
VarInt16

    End If



    If (txtInt32.Text <> "")
Then
```

```
        VarInt32 =
CLng(txtInt32.Text)

        Call
TcClientSync.WriteInt32(&H4020, 6, VarInt32)

        txtInt32.Text =
VarInt32

    End If



    If (txtReal32.Text <> "")
Then

        VarReal32 =
CSng(txtReal32.Text)

        Call
TcClientSync.WriteReal32(&H4020, 10, VarReal32)

        txtReal32.Text =
VarReal32

    End If



    If (txtReal64.Text <> "")
Then

        VarReal64 =
CDbl(txtReal64.Text)

        Call
TcClientSync.WriteReal64(&H4020, 14, VarReal64)

        txtReal64.Text =
VarReal64

    End If

End Sub


Private Sub cmdWriteByName_Click()

    Dim VarBool As Boolean

    Dim VarInt8 As Byte

    Dim VarInt16 As Integer

    Dim VarInt32 As Long

    Dim VarReal32 As Single

    Dim VarReal64 As Double



    If (txtBoolean.Text <> "")
Then

        VarBool =
CBool(txtBoolean.Text)

        Call
TcClientSync.WriteBoolVar(".PLCBoolVar", VarBool)

        txtBoolean.Text =
VarBool

    End If



    If (txtInt8.Text <> "")
```

```
Then

        VarInt8 =
CByte(txtInt8.Text)

        Call
TcClientSync.WriteInt8Var(".PLCSIntVar", VarInt8)

        txtInt8.Text =
VarInt8

    End If



    If (txtInt16.Text <> "")
Then

        VarInt16 =
CInt(txtInt16.Text)

        Call
TcClientSync.WriteInt16Var(".PLCIntVar", VarInt16)

        txtInt16.Text =
VarInt16

    End If



    If (txtInt32.Text <> "")
Then

        VarInt32 =
CLng(txtInt32.Text)

        Call
TcClientSync.WriteInt32Var(".PLCDIntVar", VarInt32)

        txtInt32.Text =
VarInt32

    End If



    If (txtReal32.Text <> "")
Then

        VarReal32 =
CSng(txtReal32.Text)

        Call
TcClientSync.WriteReal32Var(".PLCRealVar",
VarReal32)

        txtReal32.Text =
VarReal32

    End If



    If (txtReal64.Text <> "")
Then

        VarReal64 =
CDbl(txtReal64.Text)

        Call
TcClientSync.WriteReal64Var(".PLCLRealVar",
VarReal64)

        txtReal64.Text =
VarReal64

    End If

End Sub
```

```
Private Sub cmdReadState_Click()

    lblAdsState.Caption =
TcClientSync.ReadAdsState()

End Sub


Private Sub cmdWriteState_Click()

    If (txtAdsState.Text <> "")
Then

        If (txtAdsState.Text
<> TcClientSync.ReadAdsState()) Then


Call TcClientSync.WriteAdsState(txtAdsState.Text)

        End If

    End If

End Sub
```

PLC program

```
VAR_GLOBAL
    PLCBoolVar      AT %MX0.0: BOOL := TRUE;
    PLCSIntVar      AT %MB2 : USINT := 75;
    PLCIntVar       AT %MB4 : UINT := 12423;
    PLCDIntVar      AT %MB6 : UDINT := 43234532;
    PLCRealVar      AT %MB10    : REAL := 3.1415;
    PLCLRealVar     AT %MB14    : LREAL := 3.141592654;
END_VAR
```

Language / IDE	Unpack sample program
Visual Basic 6	https://infosys.beckhoff.com/content/1033/ tcscriptdll/ Resources/12459720971/.zip

4.3 Active Server Pages

Active Server Pages (ASP) provide a facility for the creation of HTML pages on the HTTP server dynamically, i.e. in response to events. This is achieved by embedding script segments in the HTML code. The scripts are then executed by the web server. TcScript.dll is used for access to TwinCAT. The following sample uses the Internet Information Server (IIS) or the Personal Web Server (PWS) from Microsoft. Creating a TcScript.dll instance can be done most easily in the global.asa. The created instance is valid within the web application for all ASP pages.

global.asa:

```
<OBJECT RUNAT="Server" SCOPE="Application" ID="TcPLC" PROGID="TcScript.TcScriptSync"> </OBJECT>
<SCRIPT LANGUAGE="VBScript" RUNAT="Server">

Sub Application_OnStart()
  Call TcPLC.ConnectTo("", 801)
End Sub

</SCRIPT>
```

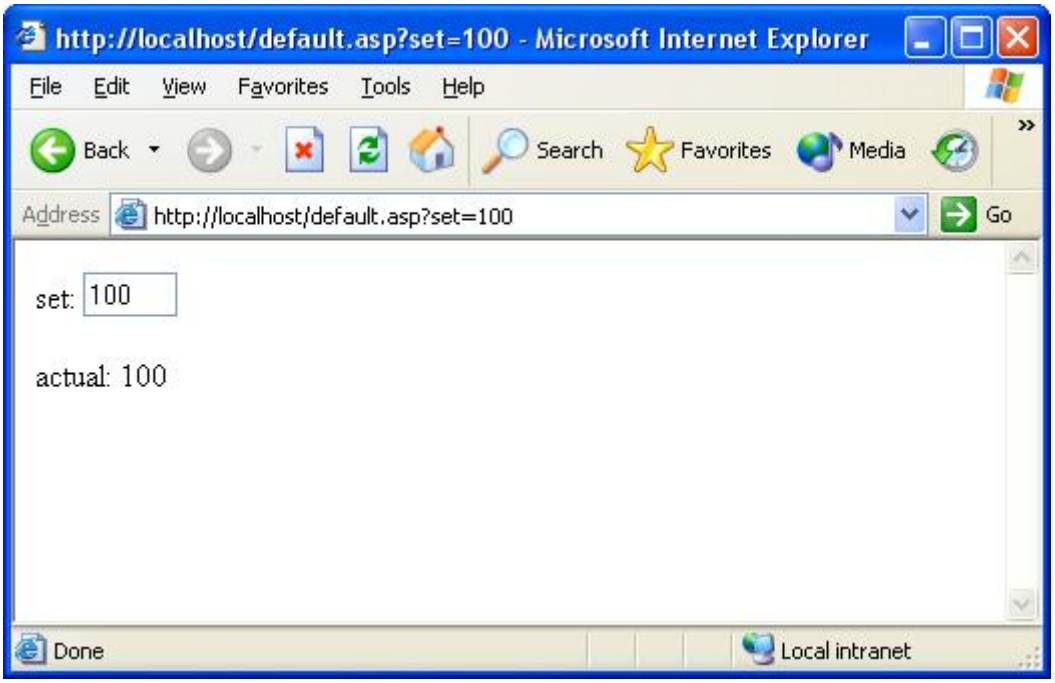
Parameters can be appended to the URL of an ASP page. These parameters are separated from the URL by a '?', and have the structure *VarName=VarWert.* At the beginning of an ASP page a check is made for whether the parameter *set* has been passed, and whether the value can be converted into a number. In that case, the parameter is copied to the VBScript variable *intSet*, and then passed to the global PLC variable *PLCVarInt*. The PLC variable is read at the end of the script, and written into the VBScript variable *intActual*.

The PLC variable is also read if no parameter was passed to the ASP page.
Within the HTML area the value of the VBScript variable *intActual* is inserted into the HTML code via the response object (provided by ASP) and transferred to the client. I.e. every time the ASP page is called (with or without parameters), the current value of the PLC variable is displayed in the explorer.
To accept input from the operator, forms are used within HTML. These entries are passed as parameters to a particular page. In our sample, the same page (default.asp) is called again. *set* is passed as a parameter (name of the input field), with the corresponding value entered by the operator. Since the page is being called again with the parameter *set*, the value following the *set* parameter is written into the PLC.

default.asp:

```
<%@ Language=VBScript %>
<%Option Explicit
Dim intActual, intSet
Application.Lock
If Not (IsEmpty(Request.Querystring("set"))) Then
  If (IsNumeric(Request.Querystring("set"))) Then
    intSet = cint(Request.Querystring("set"))
    call TcPLC.WriteVar(".PLCVarInt", intSet)
  End if
End If
intActual = TcPLC.ReadVar(".PLCVarInt")
Application.Unlock%>

<html>
<head><meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"></head>
<body>
<form method="get" action="default.asp" name="tempSet">set: <input name="set" size="4" value="<% Res
ponse.Write intSet %>"></form>
actual: <% Response.Write intActual %>
</body>
</html>
```

Presentation in Internet Explorer:



Pure HTML is sent back here to the client (Internet Explorer). It is also possible for the pages to be called by computers on which TwinCAT is not installed (e.g. a handheld PC, or even by smartphones). TwinCAT is here exclusively accessed by web servers.

The HTML page sent to the client (Internet Explorer):

```
<html>
  <head><meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"></head>
  <body>
    <form method="get" action="default.asp" name="tempSet">set: <input name="set" size="4" value="10
0"></form>
actual: 100
  </body>
</html>
```

Unpack sample program https://infosys.beckhoff.com/content/1033/ tcscriptdll/Resources/12459711755/.zip, 'Active Server Pages'.

4.4 Active Server Pages for Windows CE

This sample is a special form of Sample 3 [▶ 36]. Therefore, please first familiarize yourself with the methodology described in Sample 3 [▶ 36].

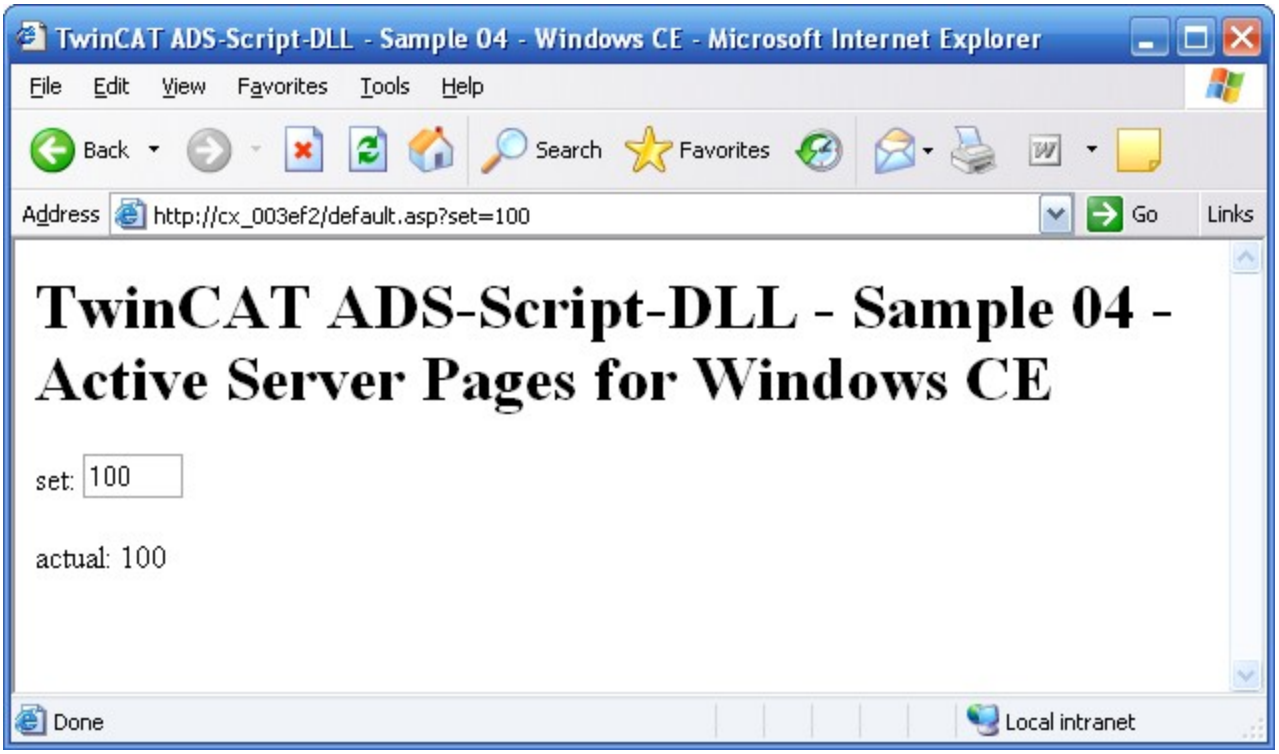
Windows CE can serve as a web server for ASP pages (by copying the appropriate files into the "WWW" directory). However, the functionality of this is limited, which manifests itself in the fact that on Windows CE the "*global.asa*" file cannot be processed. Thus, among other things, it is not possible to create global variables and objects; they must be created anew for each page call by instantiating them directly in the script (as shown below).

default.asp:

```
<%@ language=JScript %>
<%
   TcClient = new ActiveXObject("TcScript.TcScriptSync");
   TcClient.ConnectTo("", 801);

   var nIntSet = 0;
   var nIntActual = 0;
   var sSet = Request.QueryString("set");

   if ((sSet != null) && (sSet.length != 0))
   {
      var bSetIsNumber = true;

      for (var i = 0; ((i < sSet.length) && (bSetIsNumber == true)); i++)
      {
     var sChar = sSet.charAt(i);
     if (ValidChars.indexOf("0123456789.") == -1)
     {
        bSetIsNumber = false;
     }
      }

      if (bSetIsNumber == true)
      {
     nIntSet = parseInt(sSet);
     if(nIntSet >= 0)
     {
        TcClient.WriteVar(".PLCVarInt", nIntSet);
     }
      }
   }

   nIntActual = TcClient.ReadVar(".PLCVarInt");
%>

<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
      <title>TwinCAT ADS-Script-DLL - Sample 04 - Windows CE</title>
   </head>

   <body>
      <h1>TwinCAT ADS-Script-DLL - Sample 04 - Active Server Pages for Windows CE</h1>
      <form method="get" action="default.asp" name="tempSet">
     set: <input name="set" size="4" value="<% Response.Write( nIntSet ); %>">
```

```
      </form>
      actual: <% Response.Write( nIntActual ); %>
   </body>
</html>
```

Presentation in Internet Explorer:



The HTML page sent to the client (Internet Explorer):

```
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
      <title>TwinCAT ADS-Script-DLL - Sample 04 - Windows CE</title>
   </head>

   <body>
      <h1>TwinCAT ADS-Script-DLL - Sample 04 - Active Server Pages for Windows CE</h1>
      <form method="get" action="Sample04.asp" name="tempSet">
     set: <input name="set" size="4" value="3">
      </form>
      actual: 3
   </body>
</html>
```

Unpack sample program https://infosys.beckhoff.com/content/1033/ tcscriptdll/Resources/12459724811/.zip 'Active Server Pages for Windows CE'.

5 ADS Return Codes

Grouping of error codes:
Global error codes: ADS Return Codes [▶ 40]... (0x9811_0000 ...)
Router error codes: ADS Return Codes [▶ 40]... (0x9811_0500 ...)
General ADS errors: ADS Return Codes [▶ 41]... (0x9811_0700 ...)
RTime error codes: ADS Return Codes [▶ 42]... (0x9811_1000 ...)

Global error codes

Hex	Dec	HRESULT	Name	Description
0x0	0	0x98110000	ERR_NOERROR	No error.
0x1	1	0x98110001	ERR_INTERNAL	Internal error.
0x2	2	0x98110002	ERR_NORTIME	No real time.
0x3	3	0x98110003	ERR_ALLOCLOCKEDMEM	Allocation locked – memory error.
0x4	4	0x98110004	ERR_INSERTMAILBOX	Mailbox full – the ADS message could not be sent. Reducing the number of ADS messages per cycle will help.
0x5	5	0x98110005	ERR_WRONGRECEIVEHMSG	Wrong HMSG.
0x6	6	0x98110006	ERR_TARGETPORTNOTFOUND	Target port not found – ADS server is not started or is not reachable.
0x7	7	0x98110007	ERR_TARGETMACHINENOTFOUND	Target computer not found – AMS route was not found.
0x8	8	0x98110008	ERR_UNKNOWNCMDID	Unknown command ID.
0x9	9	0x98110009	ERR_BADTASKID	Invalid task ID.
0xA	10	0x9811000A	ERR_NOIO	No IO.
0xB	11	0x9811000B	ERR_UNKNOWNAMSCMD	Unknown AMS command.
0xC	12	0x9811000C	ERR_WIN32ERROR	Win32 error.
0xD	13	0x9811000D	ERR_PORTNOTCONNECTED	Port not connected.
0xE	14	0x9811000E	ERR_INVALIDAMSLENGTH	Invalid AMS length.
0xF	15	0x9811000F	ERR_INVALIDAMSNETID	Invalid AMS Net ID.
0x10	16	0x98110010	ERR_LOWINSTLEVEL	Installation level is too low –TwinCAT 2 license error.
0x11	17	0x98110011	ERR_NODEBUGINTAVAILABLE	No debugging available.
0x12	18	0x98110012	ERR_PORTDISABLED	Port disabled – TwinCAT system service not started.
0x13	19	0x98110013	ERR_PORTALREADYCONNECTED	Port already connected.
0x14	20	0x98110014	ERR_AMSSYNC_W32ERROR	AMS Sync Win32 error.
0x15	21	0x98110015	ERR_AMSSYNC_TIMEOUT	AMS Sync Timeout.
0x16	22	0x98110016	ERR_AMSSYNC_AMSERROR	AMS Sync error.
0x17	23	0x98110017	ERR_AMSSYNC_NOINDEXINMAP	No index map for AMS Sync available.
0x18	24	0x98110018	ERR_INVALIDAMSPORT	Invalid AMS port.
0x19	25	0x98110019	ERR_NOMEMORY	No memory.
0x1A	26	0x9811001A	ERR_TCPSEND	TCP send error.
0x1B	27	0x9811001B	ERR_HOSTUNREACHABLE	Host unreachable.
0x1C	28	0x9811001C	ERR_INVALIDAMSFRAGMENT	Invalid AMS fragment.
0x1D	29	0x9811001D	ERR_TLSSEND	TLS send error – secure ADS connection failed.
0x1E	30	0x9811001E	ERR_ACCESSDENIED	Access denied – secure ADS access denied.

Router error codes

Hex	Dec	HRESULT	Name	Description
0x500	1280	0x98110500	ROUTERERR_NOLOCKEDMEMORY	Locked memory cannot be allocated.
0x501	1281	0x98110501	ROUTERERR_RESIZEMEMORY	The router memory size could not be changed.
0x502	1282	0x98110502	ROUTERERR_MAILBOXFULL	The mailbox has reached the maximum number of possible messages.
0x503	1283	0x98110503	ROUTERERR_DEBUGBOXFULL	The Debug mailbox has reached the maximum number of possible messages.
0x504	1284	0x98110504	ROUTERERR_UNKNOWNPORTTYPE	The port type is unknown.
0x505	1285	0x98110505	ROUTERERR_NOTINITIALIZED	The router is not initialized.
0x506	1286	0x98110506	ROUTERERR_PORTALREADYINUSE	The port number is already assigned.

Hex	Dec	HRESULT	Name	Description
0x507	1287	0x98110507	ROUTERERR_NOTREGISTERED	The port is not registered.
0x508	1288	0x98110508	ROUTERERR_NOMOREQUEUES	The maximum number of ports has been reached.
0x509	1289	0x98110509	ROUTERERR_INVALIDPORT	The port is invalid.
0x50A	1290	0x9811050A	ROUTERERR_NOTACTIVATED	The router is not active.
0x50B	1291	0x9811050B	ROUTERERR_FRAGMENTBOXFULL	The mailbox has reached the maximum number for fragmented messages.
0x50C	1292	0x9811050C	ROUTERERR_FRAGMENTTIMEOUT	A fragment timeout has occurred.
0x50D	1293	0x9811050D	ROUTERERR_TOBEREMOVED	The port is removed.

General ADS error codes

Hex	Dec	HRESULT	Name	Description
0x700	1792	0x98110700	ADSERR_DEVICE_ERROR	General device error.
0x701	1793	0x98110701	ADSERR_DEVICE_SRVNOTSUPP	Service is not supported by the server.
0x702	1794	0x98110702	ADSERR_DEVICE_INVALIDGRP	Invalid index group.
0x703	1795	0x98110703	ADSERR_DEVICE_INVALIDOFFSET	Invalid index offset.
0x704	1796	0x98110704	ADSERR_DEVICE_INVALIDACCESS	Reading or writing not permitted.
0x705	1797	0x98110705	ADSERR_DEVICE_INVALIDSIZE	Parameter size not correct.
0x706	1798	0x98110706	ADSERR_DEVICE_INVALIDDATA	Invalid data values.
0x707	1799	0x98110707	ADSERR_DEVICE_NOTREADY	Device is not ready to operate.
0x708	1800	0x98110708	ADSERR_DEVICE_BUSY	Device is busy.
0x709	1801	0x98110709	ADSERR_DEVICE_INVALIDCONTEXT	Invalid operating system context. This can result from use of ADS blocks in different tasks. It may be possible to resolve this through multitasking synchronization in the PLC.
0x70A	1802	0x9811070A	ADSERR_DEVICE_NOMEMORY	Insufficient memory.
0x70B	1803	0x9811070B	ADSERR_DEVICE_INVALIDPARM	Invalid parameter values.
0x70C	1804	0x9811070C	ADSERR_DEVICE_NOTFOUND	Not found (files, ...).
0x70D	1805	0x9811070D	ADSERR_DEVICE_SYNTAX	Syntax error in file or command.
0x70E	1806	0x9811070E	ADSERR_DEVICE_INCOMPATIBLE	Objects do not match.
0x70F	1807	0x9811070F	ADSERR_DEVICE_EXISTS	Object already exists.
0x710	1808	0x98110710	ADSERR_DEVICE_SYMBOLNOTFOUND	Symbol not found.
0x711	1809	0x98110711	ADSERR_DEVICE_SYMBOLVERSIONINVALID	Invalid symbol version. This can occur due to an online change. Create a new handle.
0x712	1810	0x98110712	ADSERR_DEVICE_INVALIDSTATE	Device (server) is in invalid state.
0x713	1811	0x98110713	ADSERR_DEVICE_TRANSMODENOTSUPP	AdsTransMode not supported.
0x714	1812	0x98110714	ADSERR_DEVICE_NOTIFYHNDINVALID	Notification handle is invalid.
0x715	1813	0x98110715	ADSERR_DEVICE_CLIENTUNKNOWN	Notification client not registered.
0x716	1814	0x98110716	ADSERR_DEVICE_NOMOREHDLS	No further handle available.
0x717	1815	0x98110717	ADSERR_DEVICE_INVALIDWATCHSIZE	Notification size too large.
0x718	1816	0x98110718	ADSERR_DEVICE_NOTINIT	Device not initialized.
0x719	1817	0x98110719	ADSERR_DEVICE_TIMEOUT	Device has a timeout.
0x71A	1818	0x9811071A	ADSERR_DEVICE_NOINTERFACE	Interface query failed.
0x71B	1819	0x9811071B	ADSERR_DEVICE_INVALIDINTERFACE	Wrong interface requested.
0x71C	1820	0x9811071C	ADSERR_DEVICE_INVALIDCLSID	Class ID is invalid.
0x71D	1821	0x9811071D	ADSERR_DEVICE_INVALIDOBJID	Object ID is invalid.
0x71E	1822	0x9811071E	ADSERR_DEVICE_PENDING	Request pending.
0x71F	1823	0x9811071F	ADSERR_DEVICE_ABORTED	Request is aborted.
0x720	1824	0x98110720	ADSERR_DEVICE_WARNING	Signal warning.
0x721	1825	0x98110721	ADSERR_DEVICE_INVALIDARRAYIDX	Invalid array index.
0x722	1826	0x98110722	ADSERR_DEVICE_SYMBOLNOTACTIVE	Symbol not active.
0x723	1827	0x98110723	ADSERR_DEVICE_ACCESSDENIED	Access denied.
0x724	1828	0x98110724	ADSERR_DEVICE_LICENSENOTFOUND	Missing license.
0x725	1829	0x98110725	ADSERR_DEVICE_LICENSEEXPIRED	License expired.
0x726	1830	0x98110726	ADSERR_DEVICE_LICENSEEXCEEDED	License exceeded.
0x727	1831	0x98110727	ADSERR_DEVICE_LICENSEINVALID	Invalid license.
0x728	1832	0x98110728	ADSERR_DEVICE_LICENSESYSTEMID	License problem: System ID is invalid.
0x729	1833	0x98110729	ADSERR_DEVICE_LICENSENOTIMELIMIT	License not limited in time.
0x72A	1834	0x9811072A	ADSERR_DEVICE_LICENSEFUTUREISSUE	Licensing problem: time in the future.
0x72B	1835	0x9811072B	ADSERR_DEVICE_LICENSETIMETOLONG	License period too long.

Hex	Dec	HRESULT	Name	Description
0x72C	1836	0x9811072C	ADSERR_DEVICE_EXCEPTION	Exception at system startup.
0x72D	1837	0x9811072D	ADSERR_DEVICE_LICENSEDUPLICATED	License file read twice.
0x72E	1838	0x9811072E	ADSERR_DEVICE_SIGNATUREINVALID	Invalid signature.
0x72F	1839	0x9811072F	ADSERR_DEVICE_CERTIFICATEINVALID	Invalid certificate.
0x730	1840	0x98110730	ADSERR_DEVICE_LICENSEOEMNOTFOUND	Public key not known from OEM.
0x731	1841	0x98110731	ADSERR_DEVICE_LICENSERESTRICTED	License not valid for this system ID.
0x732	1842	0x98110732	ADSERR_DEVICE_LICENSEDEMODENIED	Demo license prohibited.
0x733	1843	0x98110733	ADSERR_DEVICE_INVALIDFNCID	Invalid function ID.
0x734	1844	0x98110734	ADSERR_DEVICE_OUTOFRANGE	Outside the valid range.
0x735	1845	0x98110735	ADSERR_DEVICE_INVALIDALIGNMENT	Invalid alignment.
0x736	1846	0x98110736	ADSERR_DEVICE_LICENSEPLATFORM	Invalid platform level.
0x737	1847	0x98110737	ADSERR_DEVICE_FORWARD_PL	Context – forward to passive level.
0x738	1848	0x98110738	ADSERR_DEVICE_FORWARD_DL	Context – forward to dispatch level.
0x739	1849	0x98110739	ADSERR_DEVICE_FORWARD_RT	Context – forward to real time.
0x740	1856	0x98110740	ADSERR_CLIENT_ERROR	Client error.
0x741	1857	0x98110741	ADSERR_CLIENT_INVALIDPARM	Service contains an invalid parameter.
0x742	1858	0x98110742	ADSERR_CLIENT_LISTEMPTY	Polling list is empty.
0x743	1859	0x98110743	ADSERR_CLIENT_VARUSED	Var connection already in use.
0x744	1860	0x98110744	ADSERR_CLIENT_DUPLINVOKEID	The called ID is already in use.
0x745	1861	0x98110745	ADSERR_CLIENT_SYNCTIMEOUT	Timeout has occurred – the remote terminal is not responding in the specified ADS timeout. The route setting of the remote terminal may be configured incorrectly.
0x746	1862	0x98110746	ADSERR_CLIENT_W32ERROR	Error in Win32 subsystem.
0x747	1863	0x98110747	ADSERR_CLIENT_TIMEOUTINVALID	Invalid client timeout value.
0x748	1864	0x98110748	ADSERR_CLIENT_PORTNOTOPEN	Port not open.
0x749	1865	0x98110749	ADSERR_CLIENT_NOAMSADDR	No AMS address.
0x750	1872	0x98110750	ADSERR_CLIENT_SYNCINTERNAL	Internal error in Ads sync.
0x751	1873	0x98110751	ADSERR_CLIENT_ADDHASH	Hash table overflow.
0x752	1874	0x98110752	ADSERR_CLIENT_REMOVEHASH	Key not found in the table.
0x753	1875	0x98110753	ADSERR_CLIENT_NOMORESYM	No symbols in the cache.
0x754	1876	0x98110754	ADSERR_CLIENT_SYNCRESINVALID	Invalid response received.
0x755	1877	0x98110755	ADSERR_CLIENT_SYNCPORTLOCKED	Sync Port is locked.

RTime error codes

Hex	Dec	HRESULT	Name	Description
0x1000	4096	0x98111000	RTERR_INTERNAL	Internal error in the real-time system.
0x1001	4097	0x98111001	RTERR_BADTIMERPERIODS	Timer value is not valid.
0x1002	4098	0x98111002	RTERR_INVALIDTASKPTR	Task pointer has the invalid value 0 (zero).
0x1003	4099	0x98111003	RTERR_INVALIDSTACKPTR	Stack pointer has the invalid value 0 (zero).
0x1004	4100	0x98111004	RTERR_PRIOEXISTS	The request task priority is already assigned.
0x1005	4101	0x98111005	RTERR_NOMORETCB	No free TCB (Task Control Block) available. The maximum number of TCBs is 64.
0x1006	4102	0x98111006	RTERR_NOMORESEMAS	No free semaphores available. The maximum number of semaphores is 64.
0x1007	4103	0x98111007	RTERR_NOMOREQUEUES	No free space available in the queue. The maximum number of positions in the queue is 64.
0x100D	4109	0x9811100D	RTERR_EXTIRQALREADYDEF	An external synchronization interrupt is already applied.
0x100E	4110	0x9811100E	RTERR_EXTIRQNOTDEF	No external sync interrupt applied.
0x100F	4111	0x9811100F	RTERR_EXTIRQINSTALLFAILED	Application of the external synchronization interrupt has failed.
0x1010	4112	0x98111010	RTERR_IRQLNOTLESSOREQUAL	Call of a service function in the wrong context
0x1017	4119	0x98111017	RTERR_VMXNOTSUPPORTED	Intel VT-x extension is not supported.
0x1018	4120	0x98111018	RTERR_VMXDISABLED	Intel VT-x extension is not enabled in the BIOS.
0x1019	4121	0x98111019	RTERR_VMXCONTROLSMISSING	Missing function in Intel VT-x extension.
0x101A	4122	0x9811101A	RTERR_VMXENABLEFAILS	Activation of Intel VT-x fails.

Specific positive HRESULT Return Codes:

HRESULT	Name	Description
0x0000_0000	S_OK	No error.
0x0000_0001	S_FALSE	No error. Example: successful processing, but with a negative or incomplete result.
0x0000_0203	S_PENDING	No error. Example: successful processing, but no result is available yet.
0x0000_0256	S_WATCHDOG_TIMEOUT	No error. Example: successful processing, but a timeout occurred.

TCP Winsock error codes

Hex	Dec	Name	Description
0x274C	10060	WSAETIMEDOUT	A connection timeout has occurred - error while establishing the connection, because the remote terminal did not respond properly after a certain period of time, or the established connection could not be maintained because the connected host did not respond.
0x274D	10061	WSAECONNREFUSED	Connection refused - no connection could be established because the target computer has explicitly rejected it. This error usually results from an attempt to connect to a service that is inactive on the external host, that is, a service for which no server application is running.
0x2751	10065	WSAEHOSTUNREACH	No route to host - a socket operation referred to an unavailable host.
More Winsock error codes: Win32 error codes			

More Information:
www.beckhoff.com/automation

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
Phone: +49 5246 9630
info@beckhoff.com
www.beckhoff.com