BECKHOFF New Automation Technology

Manual | EN

TS6350-0030

TwinCAT 2 | SMS/SMTP Server CE

Supplement | Communication

2023-01-06 | Version: 1.1

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

© 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

SMTP Server:

The TwinCAT SMTP Server is used to send E-Mail messages with TwinCAT via ADS. Additional information can be find here [▶ 35].

Encryption:

Since TC SMTP server version 1.0.14 SSL/STARTTLS encryption is supported

LF images (low footprint e.g.: CX9000) do not support encryption
Please use Version 1.0.13 Windows CE image.

3 SMS Server

The TwinCAT SMS Server is used to send SMS messages with TwinCAT via a GSM modem.

Two product components are available for this:

1. TwinCAT PLC Libraries: SMS / SMS BC [▶ 9] (sends SMS messages directly from the PLC)
2. TwinCAT ADS Device: SMS COM Server [▶ 17] (sends SMS messages via ADS, e.g. from a Visual Basic program)

3.1 PLC libraries

The TwinCAT SMS library contains a block for sending SMS messages directly from the PLC. The SMS library is based on the 'Serial Communication' (COMlib) library. This makes it possible to communicate with the PC's serial interface and with the serial terminal (KL6xxx) in the same way. More detailed information may be found in the documentation for the ‘Serial Communication' library.

Properties:

- Sending an SMS via a connected GSM modem.
- The devices listed below can be connected via a serial data cable to the serial interface of the TwinCAT PC or via the Kl6xxx serial terminal.
- Sending SMS messages of up to 160 characters.
- It is available for the PC and for the Bus Controller (BC).

Requirements:

- Installed TwinCAT System, Installation level: TwinCAT PLC or higher.
- TwinCAT PLC runtime on PC or BC (Bus Controller).
- Suitable GSM modem with a data cable.

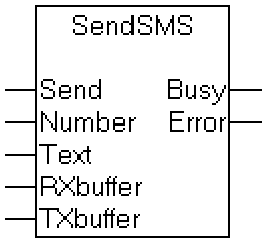
Currently supported devices:

- Westermo GS-01 (communication parameters: 9600 baud, 8 data bits, no parity bit, one stop bit)
- Siemens S35i (communication parameters: 19200 baud, 8 data bits, no parity bit, one stop bit)
- Nokia 6210 (communication parameters: 19200 baud, 8 data bits, no parity bit, one stop bit)
- Maestro 100 (communication parameters: 9600 baud, 8 data bits, no parity bit, one stop bit)

The following files are copied into the ..\TwinCAT\PLC\Lib directory during installation:

- TcPlcSMS.Lib (SMS library for the PC)
- TcPlcSMSBC.lb6 (SMS library for the BC)
- COMlib.lib, COMlibBC5B.lb6, COMlibBCext.lb6, ChrAsc.Lib and ChrAsc.obj (Serial Communication libraries for the PC and the BC)
- COMlibV2.lib, COMlibV2lb6 (Serial Communication library for the PC and the Bus Controller (BC) v2.0). Only with TwinCAT SMS Server version 2.0 and higher!

3.1.1 SendSMS



The **SendSMS** function block allows an SMS to be sent via a connected GSM modem. The function block is based on the 'Serial Communication' library.

Because the block only communicates via the **ComBuffer** structure in the 'Serial Communication' library, instances can be formed, and it can be applied to every kind of serial interface.

VAR_INPUT

```
Send         : BOOL;
Number       : String;
Text         : String(160);
```

Send: The function block is activated by a positive edge at this input.

Number: telephone number to be dialled in national format (e.g.: 0170123456)

Text: The SMS message to be sent

VAR_OUTPUT

```
Busy          : BOOL;
Error        : INT;
```

Busy: This output is set when there is a rising edge at the Send input, and remains set until the SMS has been sent to the modem or until an error has occurred.

Error: If an error occurs while the SMS is being transferred, the Busy output is reset, and an error code is made available at the Error output. If the Error output is 0, the transfer was successful.

The function block can return the following errors:

Number	Meaning	Cause
1	Communication with the modem is not possible.	Is the terminal correctly configured? Has the appropriate ComLib library been used?
2	Modem reports an error during configuration.	Is a compatible GSM modem connected?
3	Modem cannot send SMS.	Is the SIM card working properly? Can the card be used without entering the PIN? Is the modem connected to the network? Is a compatible modem connected?
4	Communication error.	Has the correct transmission speed been set?

VAR_IN_OUT

```
RXbuffer      : ComBuffer;
TXbuffer      : ComBuffer;
```

RXbuffer: Structure for communication with the serial interface. An interface-specific block in the 'Serial Communication' library fills this buffer with the data for the interface.

TXbuffer: Structure for communication with the serial interface. An interface-specific block in the 'Serial Communication' library transfers the data from this buffer to the interface.

These structures, and their usage, are described in more detail in the documentation for the 'Serial Communication' library. The SendSMS block is here connected to a SendString or ReceiveString block.

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v2.7.0 and higher	PC (i386)	TcPlcSMS.Lib, ChrAsc.Lib, COMLib.Lib, Standard.Lib, PlcHelper.Lib
TwinCAT v2.8.0 and higher	PC (i386)	TcPlcSMS.Lib, ChrAsc.Lib, COMLib.Lib, TcSystem.Lib, (Standard.Lib; TcBase.Lib; are included automatically)
TwinCAT v2.7.0 and higher	BCxxxx (165)	TcPlcSMSBC.Lb6, Standard.Lb6, PlcHelperBC.Lb6, ChrAsc.Lb6, COMLibBC5B.Lb6

3.1.2 Get_TcPlcSMS_Version

Get_TcPlcSMS_Version
bGet

The function returns library version info.

FUNCTION Get_TcPlcSMS_Version: STRING(20)

VAR_INPUT

```
bGet : BOOL;
```

bGet: The compiler requires at least one input parameter for functions. You can set this parameter to TRUE or FALSE.

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v2.7.0 and higher	PC (i386)	TcPlcSMS.Lib, ChrAsc.Lib, COMLib.Lib, Standard.Lib, PlcHelper.Lib
TwinCAT v2.8.0 and higher	PC (i386)	TcPlcSMS.Lib, ChrAsc.Lib, COMLib.Lib, TcSystem.Lib, (Standard.Lib; TcBase.Lib; are included automatically)

3.1.3 Get_TcPlsSMSBC_Version

Get_TcPlcSMSBC_Version
bGet

The function returns library version info.

FUNCTION Get_TcPlcSMSBC_Version: STRING(20)

VAR_INPUT

```
bGet : BOOL;
```

bGet:he compiler requires at least one input parameter for functions. You can set this parameter to TRUE or FALSE.

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v2.7.0 and higher	BCxxxx (165)	TcPlcSMSBC.Lb6, Standard.Lb6, PlcHelperBC.Lb6, ChrAsc.Lb6, COMLibBC5B.Lb6

3.1.4 Examples

3.1.4.1 Sending an SMS Using a Function Block in the BC via KL6001

Source text: https://infosys.beckhoff.com/content/1033/tcsmssmtpsrvce/Resources/11386387595/.exe (do not forget to change the telephone number)

Task

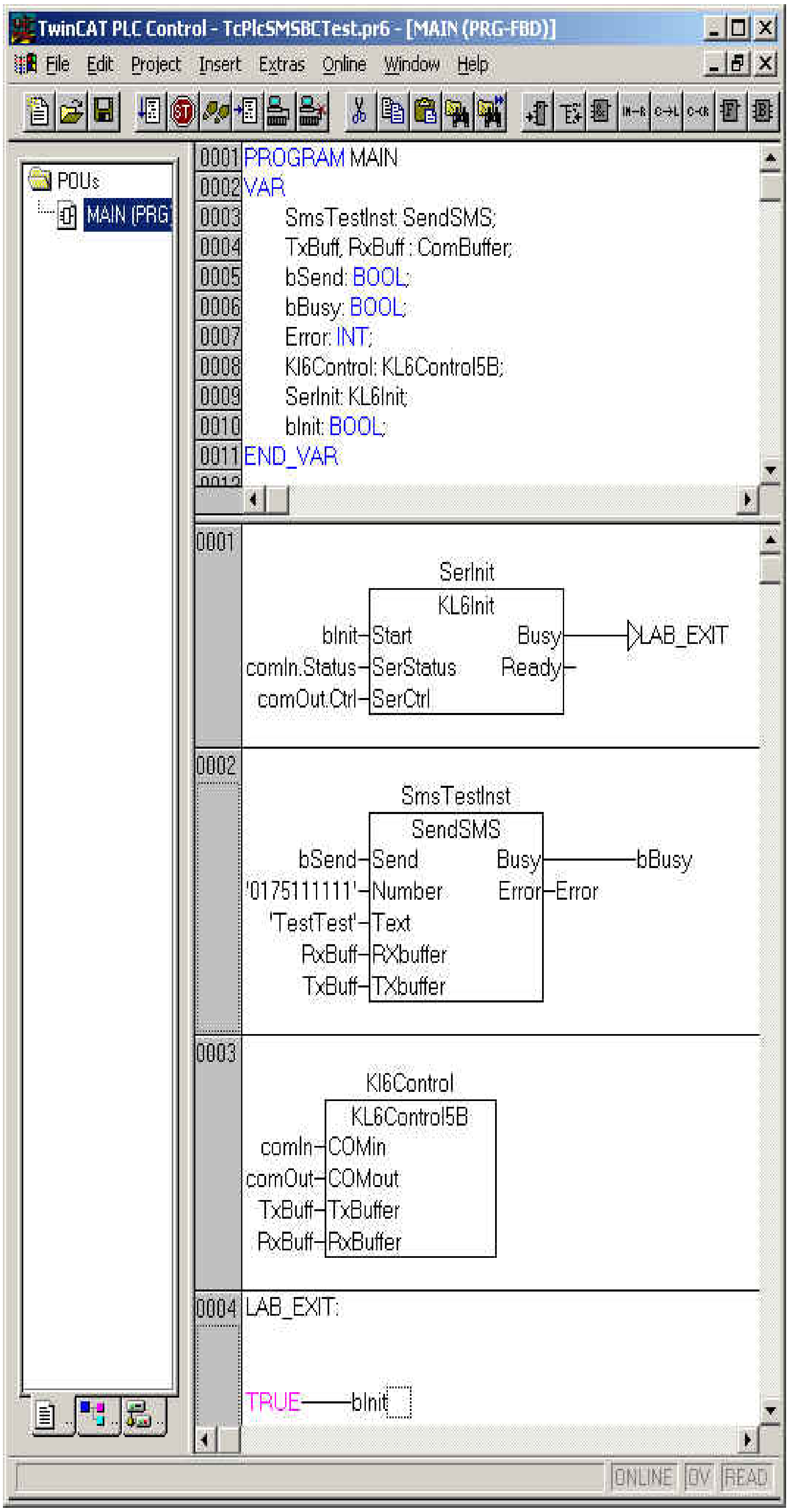
A simple program that uses the SMS function block on a BC9000 to send an SMS via the KL6001 serial terminal.

Description

The serial interface is first initialized using the KL6Init block.

It is then possible to initiate sending with a rising edge at the Send input.

Implementation


TwinCAT PLC Control - TcPlcSMSBCTest.pr6 - [MAIN (PRG-FBD)]
File Edit Project Insert Extras Online Window Help
POUs
MAIN (PRG
ONLINE OV READ

```
PROGRAM MAIN
VAR
    SmsTestInst: SendSMS;
    TxBuff, RxBuff : ComBuffer;
    bSend: BOOL;
    bBusy: BOOL;
    Error: INT;
    Kl6Control: KL6Control5B;
    SerInit: KL6Init;
    bInit: BOOL;
END_VAR
```

```
0001
                 SerInit
                 KL6Init
         bInit-Start          Busy-----LAB_EXIT
comIn.Status-SerStatus       Ready-
  comOut.Ctrl-SerCtrl

0002
                 SmsTestInst
                 SendSMS
        bSend-Send            Busy-----bBusy
'0175111111'-Number          Error-Error
   'TestTest'-Text
       RxBuff-RXbuffer
       TxBuff-TXbuffer

0003
              Kl6Control
              KL6Control5B
   comIn-COMin
  comOut-COMout
  TxBuff-TxBuffer
  RxBuff-RxBuffer

0004 LAB_EXIT:

TRUE-----bInit
```

3.1.4.2 Sending an SMS with a Function Block on the PC via the PC's Serial Interface

Source text: https://infosys.beckhoff.com/content/1033/tcsmssmtpsrvce/Resources/11386389003/.exe (do not forget to change the telephone number)

Task

A simple program that uses the SMS function block on a PC to send an SMS via its serial interface.

Description

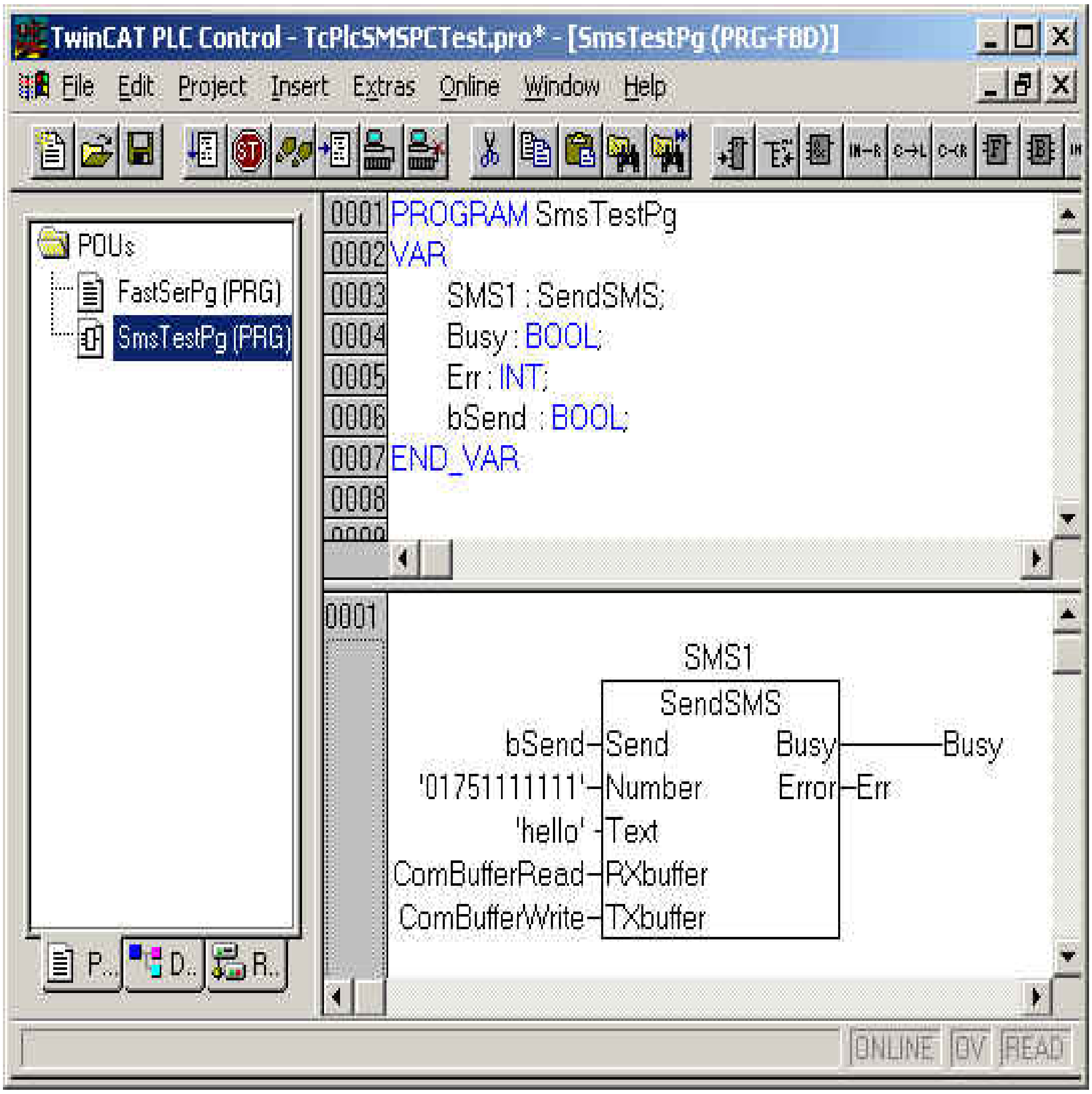
As is described in the documentation for the 'Serial Communication' library, the serial interface is served in a fast task, while the SMS message is sent by a slower task.

Sending is initiated by a rising edge at the Send input. The Busy variable can be used to detect when the SMS message has been transferred to the modem.

Configuration in the System Manager

So that the serial interface can be controlled, it must be inserted as an I/O device in the System Manager. The interface must be configured for KL6xx1 operation with 64 data bytes. The inputs and outputs of the COM port device must then be linked to the variables in the PLC program (SerInData and SerOutData).

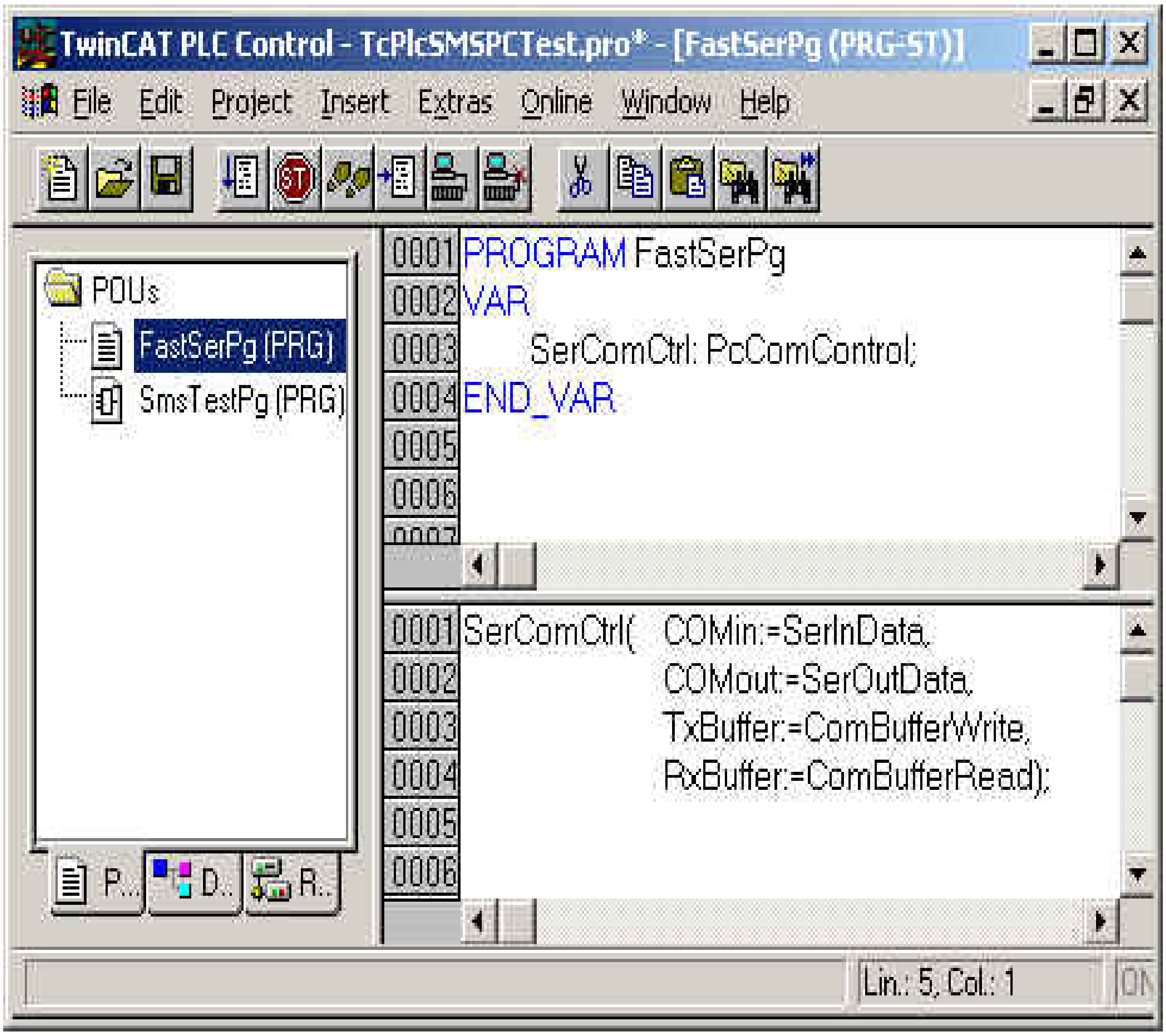
Implementation of the Program in the Slow Task to Send the SMS


TwinCAT PLC Control - TcPlcSMSPCTest.pro* - [SmsTestPg (PRG-FBD)]
File Edit Project Insert Extras Online Window Help
POUs
FastSerPg (PRG)
SmsTestPg (PRG)
P.. D.. R..
ONLINE OV READ

```
PROGRAM SmsTestPg
VAR
        SMS1 : SendSMS;
        Busy : BOOL;
        Err : INT;
        bSend  : BOOL;
END_VAR
```

```
                         SMS1
                  +-----------------+
                  |     SendSMS     |
          bSend --|Send         Busy|------Busy
  '01751111111' --|Number      Error|--Err
        'hello' --|Text             |
  ComBufferRead --|RXbuffer         |
 ComBufferWrite --|TXbuffer         |
                  +-----------------+
```

Implementation of the Program in the Fast Task for Serving the Serial Interface


TwinCAT PLC Control - TcPlcSMSPCTest.pro* - [FastSerPg (PRG-ST)]
File Edit Project Insert Extras Online Window Help
POUs
FastSerPg (PRG)
SmsTestPg (PRG)
P... D... R...
Lin.: 5, Col.: 1

```
PROGRAM FastSerPg
VAR
    SerComCtrl: PcComControl;
END_VAR
```

```
SerComCtrl(   COMin:=SerInData,
              COMout:=SerOutData,
              TxBuffer:=ComBufferWrite,
              RxBuffer:=ComBufferRead);
```

Global Variables

```
VAR_GLOBAL
    ComBufferRead : ComBuffer;
    ComBufferWrite : ComBuffer;
    SerInData AT %IB100 : PcComInData;
    SerOutData AT %QB100 : PcComOutData;
END_VAR
```

Task configuration

```
⊟····Task Configuration
     ⊟···· Standard (PRIORITY := 0, INTERVAL := T#10ms)
     ¦      └····· SmsTestPg
     ⊟···· FastSerTask (PRIORITY := 1, INTERVAL := T#1ms)
            └····· FastSerPg
```

3.2 ADS device

The TwinCAT ADS Device: SMS COM Server is a software driver that can send SMS messages over a GSM modem.

Because the SMS server is a pure software driver, it can be described as a virtual field device (automation device). For this reason it makes a Beckhoff ADS (Automation Device Specification) interface available to other communication partners (e.g. the PLC or Visual Basic programs). The use of the ADS standardises access to the TwinCAT SMS device, placing it in the group of available virtual field devices.

Properties:

- Implementation as an NT COM Service.
- A PLC program is not necessary.
- Implementation as a Beckhoff ADS device, with ADS port number 10400.
- Is started and stopped along with TwinCAT.
- The GSM modem must be connected via a serial data cable to the serial interface of the TwinCAT PC.
- The TwinCAT SMS Server Configurator sets the communication parameters for the connected GSM modem.
- Sending SMS messages of up to 160 characters.
- 50 SMS Test Version: Only a test version will be installed if a valid license key is not entered during installation. This has full functionality, but is restricted to 50 messages.

Requirements:

- Installed TwinCAT System, Installation level: TwinCAT CP or higher.
- Suitable GSM modem with a data cable.

Supported devices:

- Westermo GS-01 (COM parameters: 9600,N,8,1)
- Siemens S35i (COM parameters: 19200,N,8,1)
- Nokia 6210 (COM parameters: 19200,N,8,1)
- Maestro 100 (COM parameters: 9600,N,8,1)

The following files are copied into the ..\TwinCAT\SMS directory during installation:

- TcSmsSrv.exe (TwinCAT ADS Device: SMS COM Server).
- TcSmsSrvCfg.exe (SMS COM Server Configurator. Configures the connection with the GSM modem [▶ 17]).
- SmsSrvTest.exe (Visual Basic test application to send test SMS message).
- TcSmsSrvSetup.txt (Installation and configuration infos)

Also see about this

Configuration [▶ 17]

3.2.1 Configuration

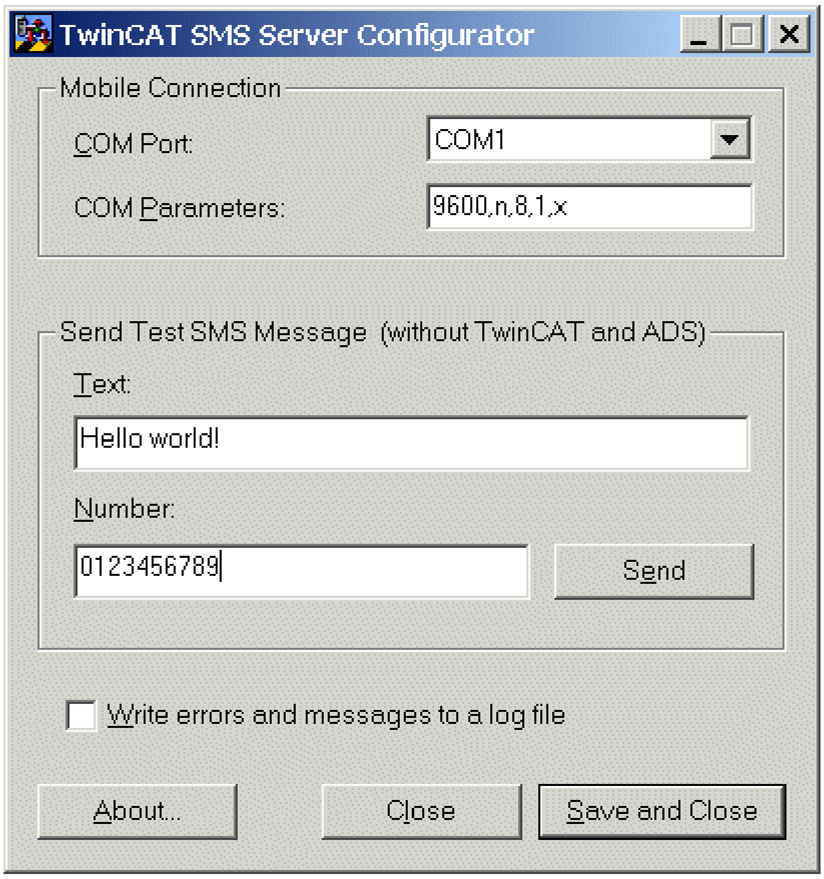
The TwinCAT SMS Server Configurator is used to configure the SMS COM Server. The Configurator allows the serial interface used and the communication parameters applied to be set. The keeping of records in a log file is also activated by means of the Configurator.

It is not necessary for a GSM modem to be connected in order to carry out the configuration, since it is possible to perform the configuration before the device starts operation. The configuration data is saved in the Default.tps file in the TwinCAT directory. This means that the configuration can be secured with this file, or can be copied to a target computer.

1. Following installation, but before the configuration itself, the SMS COM server must be registered **once** as a TwinCAT device. This requires the TwinCAT system to be halted (red icon). Make the following entry on the command line, and confirm with the return key:

C:\TwinCAT\SMS\TcSmsSrv.exe /RegTcServer

2. The actual configuration can now be carried out. It should be noted that it is also necessary to stop the TwinCAT system in order to use the configurator. Start **TcSmsSrvCfg.exe**


TwinCAT SMS Server Configurator

Mobile Connection

COM Port: COM1

COM Parameters: 9600,n,8,1,x

Send Test SMS Message (without TwinCAT and ADS)

Text: Hello world!

Number: 0123456789

Choosing the Serial Interface

The interface to which the GSM modem is connected must be selected in the 'COM Port' selection box (Device configuration string syntax [▶ 33]),

Configuring the Communication Parameters

The communication parameters applying to the serial interface must be set in accordance with the data provided by the manufacturer of the GSM modem.
In most cases, the standard setting (19200,n,8,1) should be adequate.

Switch on the Log Function

The log function can be switched on if more precise information is needed about errors, or in order to obtain a record of the messages sent.
When the log function is switched on, the file 'TcSmsSrvLog.xml' is created in the ..\TwinCAT or ..\WINDOWS\System32 directory. All messages sent, and all errors, are recorded here.

Sending a Test SMS

A test SMS should be sent to find out whether everything has been correctly set up. An easy way to do this is to use the Visual Basic example program.

Also see about this

- Device control string syntax [▶ 33]

3.2.2 ADS interface

The TwinCAT SMS Server is a software driver that can send SMS messages over a connected GSM modem. For this reason, it makes a Beckhoff ADS (Automation Device Specification) interface available to other communication partners (e.g. the PLC or Visual Basic programs). The use of the ADS standardizes access to the TwinCAT SMS device, placing it in the group of available virtual field devices.

The READ and WRITE operations to and from the interface are implemented, as specified by ADS, using two numbers: the index group and the index offset.
A more detailed description of the group and offset indexes in relation to the SMS Server's ADS interface is given on the following pages.

Specifications of the PLC's "Index Group"

The two global areas of an ADS device are represented for the SMS Server in the index groups as follows:

Index-Group (0x = hex)	Index Group description
0x00003000	Configuration area [▶ 19]
0x00004000	Area for SMS services [▶ 19]

3.2.2.1 "Index-Group/Offset" Specification for TwinCAT SMS Server Services

This section describes ADS services for sending SMS messages with the TwinCAT SMS Server.

Index group	Index offset	Ac-cess	Data type	Physical unit	Definition area	Description	Remark
0x00004000	0x00000001	W	UINT8[n]	Character string	Null-terminated character string	The XML character string describes the contents and the destination of the SMS message. [▶ 20]	

3.2.2.2 "Index-Group/Offset" Specification for TwinCAT SMS Server Configuration

This section describes ADS services for configuration of the TwinCAT SMS Server.

Index group	Index offset	Ac-cess	Data type	Physical unit	Defini-tion area	Description	Remark
0x00003000	0x00000001	W	UINT8	1	0/1	Switch log file on/off (1 = log file is written)	

3.2.2.3 Send SMS

This ADS service sends an SMS to one or more recipients.

Use the ADS service ”AdsWriteReq”.

Name	Description
ADS Port	10400
IndexGroup	0x00004000
IndexOffset	0x00000001
cbLength	Length of the XML character string, including the terminating null character.
pWriteData	XML character string with the following structure: **<Msg>** **<Targets><Target><No></No></Target></Targets>** **<Body><![CDATA[]]></Body>** **</Msg>** *Destination Message*
<No></No> *Destination*	The **<No></No>***Destination* can be repeated in order to send to multiple recipients.
Destination	**Destination** is the number to be dialled. It can be entered in national or international form. National numbers have a leading 0, as for example: 0170111111. International numbers begin with a + sign, followed by the country code, as for example: +49170111111.
Message	*Message* is the text that is to be sent. The message may not contain more than 80 characters!

Example of the XML character string:

```
<Msg>
    <Targets>
        <Target>
            <No>0170111111</No>
            <No>0170222222</No>
        </Target>
    </Targets>

    <Body>
        <![CDATA[Hello, this is an SMS]]>
    </Body>
</Msg>
```

3.2.2.4 ADS Return Codes

Error codes: 0x000... [▸ 20], 0x500... [▸ 20], 0x700... [▸ 20], 0x1000... [▸ 20], 0x274C... [▸ 20]

Global Error Codes

Hex	Dec	Description	Possible Causes	Solution
0x0	0	no error		
0x1	1	Internal error		
0x2	2	No Rtime		
0x3	3	Allocation locked memory error		
0x4	4	Insert mailbox error	No ADS mailbox was available to process this message.	Reduce the number of ADS calls (e.g ADS-Sum commands or Max Delay Parameter)
0x5	5	Wrong receive HMSG		
0x6	6	target port not found	ADS Server not started	
0x7	7	target machine not found	Missing ADS routes	

Hex	Dec	Description	Possible Causes	Solution
0x8	8	Unknown command ID		
0x9	9	Bad task ID		
0xA	10	No IO		
0xB	11	Unknown ADS command		
0xC	12	Win 32 error		
0xD	13	Port not connected		
0xE	14	Invalid ADS length		
0xF	15	Invalid AMS Net ID		
0x10	16	Low Installation level		
0x11	17	No debug available		
0x12	18	Port disabled		
0x13	19	Port already connected		
0x14	20	ADS Sync Win32 error		
0x15	21	ADS Sync Timeout		
0x16	22	ADS Sync AMS error		
0x17	23	ADS Sync no index map		
0x18	24	Invalid ADS port		
0x19	25	No memory		
0x1A	26	TCP send error		
0x1B	27	Host unreachable		
0x1C	28	Invalid AMS fragment		

Router Error Codes

Hex	Dec	Description	Possible Causes	Solution
0x500	1280	ROUTERERR_NOLOCKEDMEMORY	No locked memory can be allocated	
0x501	1281	ROUTERERR_RESIZEMEMORY	The size of the router memory could not be changed	
0x502	1282	ROUTERERR_MAILBOXFULL	The mailbox has reached the maximum number of possible messages. The current sent message was rejected	Check the connection between the communication partners
0x503	1283	ROUTERERR_DEBUGBOXFULL	The mailbox has reached the maximum number of possible messages. The sent message will not be displayed in the debug monitor	Check the connection to the debug monitor
0x504	1284	ROUTERERR_UNKNOWNPORTTYPE	The port type is unknown	
0x505	1285	ROUTERERR_NOTINITIALIZED	Router is not initialised	
0x506	1286	ROUTERERR_PORTALREADYINUSE	The desired port number is already assigned	
0x507	1287	ROUTERERR_NOTREGISTERED	Port not registered	
0x508	1288	ROUTERERR_NOMOREQUEUES	The maximum number of Ports reached	
0x509	1289	ROUTERERR_INVALIDPORT	The port is invalid.	

Hex	Dec	Description	Possible Causes	Solution
0x50A	1290	ROUTERERR_NOTACTIVATED	TwinCAT Router not active	
0x50B	1291	ROUTERERR_FRAGMENTBOXFULL		
0x50C	1292	ROUTERERR_FRAGMENTTIMEOUT		
0x50D	1293	ROUTERERR_TOBEREMOVED		

General ADS Error Codes

Hex	Dec	Description	Possible Causes	Solution
0x700	1792	error class <device error>		
0x701	1793	Service is not supported by server		
0x702	1794	invalid index group		
0x703	1795	invalid index offset		
0x704	1796	reading/writing not permitted		
0x705	1797	parameter size not correct		
0x706	1798	invalid parameter value(s)		
0x707	1799	device is not in a ready state		
0x708	1800	device is busy		
0x709	1801	invalid context (must be in Windows)		
0x70A	1802	out of memory		
0x70B	1803	invalid parameter value(s)		
0x70C	1804	not found (files, ...)		
0x70D	1805	syntax error in command or file		
0x70E	1806	objects do not match		
0x70F	1807	object already exists		
0x710	1808	symbol not found		
0x711	1809	symbol version invalid	Onlinechange	Release handle and get a new one
0x712	1810	server is in invalid state		
0x713	1811	AdsTransMode not supported		
0x714	1812	Notification handle is invalid	Onlinechange	Release handle and get a new one
0x715	1813	Notification client not registered		
0x716	1814	no more notification handles		
0x717	1815	size for watch too big		
0x718	1816	device not initialized		
0x719	1817	device has a timeout		
0x71A	1818	query interface failed		
0x71B	1819	wrong interface required		
0x71C	1820	class ID is invalid		
0x71D	1821	object ID is invalid		
0x71E	1822	request is pending		
0x71F	1823	request is aborted		
0x720	1824	signal warning		
0x721	1825	invalid array index		

Hex	Dec	Description	Possible Causes	Solution
0x722	1826	symbol not active	Onlinechange	Release handle and get a new one
0x723	1827	access denied		
0x724	1828	missing license		Activate license for TwinCAT 3 function
0x72c	1836	exception occured during system start		Check each device transistions
0x740	1856	Error class <client error>		
0x741	1857	invalid parameter at service		
0x742	1858	polling list is empty		
0x743	1859	var connection already in use		
0x744	1860	invoke ID in use		
0x745	1861	timeout elapsed		Check ADS routes of sender and receiver and your firewall setting
0x746	1862	error in win32 subsystem		
0x747	1863	Invalid client timeout value		
0x748	1864	ads-port not opened		
0x750	1872	internal error in ads sync		
0x751	1873	hash table overflow		
0x752	1874	key not found in hash		
0x753	1875	no more symbols in cache		
0x754	1876	invalid response received		
0x755	1877	sync port is locked		

RTime Error Codes

Hex	Dec	Description	Possible Causes
0x1000	4096	RTERR_INTERNAL	Internal fatal error in the TwinCAT real-time system
0x1001	4097	RTERR_BADTIMERPERIODS	Timer value not vaild
0x1002	4098	RTERR_INVALIDTASKPTR	Task pointer has the invalid value ZERO
0x1003	4099	RTERR_INVALIDSTACKPTR	Task stack pointer has the invalid value ZERO
0x1004	4100	RTERR_PRIOEXISTS	The demand task priority is already assigned
0x1005	4101	RTERR_NOMORETCB	No more free TCB (Task Control Block) available. Maximum number of TCBs is 64
0x1006	4102	RTERR_NOMORESEMAS	No more free semaphores available. Maximum number of semaphores is 64
0x1007	4103	RTERR_NOMOREQUEUES	No more free queue available. Maximum number of queue is 64
0x1008	4104	TwinCAT reserved.	
0x1009	4105	TwinCAT reserved.	
0x100A	4106	TwinCAT reserved.	
0x100B	4107	TwinCAT reserved.	
0x100C	4108	TwinCAT reserved.	
0x100D	4109	RTERR_EXTIRQALREADYDEF	An external synchronisation interrupt is already applied
0x100E	4110	RTERR_EXTIRQNOTDEF	No external synchronsiation interrupt applied
0x100F	4111	RTERR_EXTIRQINSTALLFAILED	The apply of the external synchronisation interrupt failed
0x1010	4112	RTERR_IRQLNOTLESSOREQUAL	Call of a service function in the wrong context
0x1017	4119	RTERR_VMXNOTSUPPORTED	Intel VT-x extension is not supported.

Hex	Dec	Description	Possible Causes
0x1018	4120	RTERR_VMXDISABLED	Intel VT-x extension is not enabled in BIOS.
0x1019	4121	RTERR_VMXCONTROLSMISSING	Missing feature in Intel VT-x extension.
0x101A	4122	RTERR_VMXENABLEFAILS	Enabling Intel VT-x fails.

TCP Winsock Error Codes

Hex	Dec	Description	Possible Causes	Solution
0x274c	10060	A socket operation was attempted to an unreachable host	Host unreachable	Check network connection via ping
0x274d	10061	A connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed because connected host has failed to respond.	Host unreachable	Check network connection via ping
0x2751	10065	No connection could be made because the target machine actively refused it		
		Further Winsock error codes: Win32 Error Codes		

3.2.3 Examples

3.2.3.1 Sending an SMS with Visual Basic

Source text: https://infosys.beckhoff.com/content/1033/tcsmssmtpsrvce/Resources/11386390411/.exe

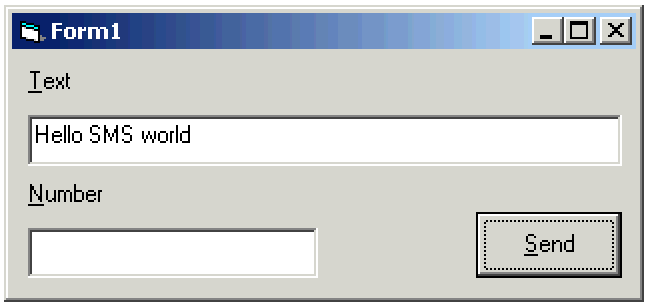
Task

A simple Visual Basic program for sending an SMS.

Description

In this example, the ADS-OCX is used to establish an ADS connection to the TwinCAT SMS Server. The SMS text and the telephone number can be entered on a form. The message is then sent by clicking a button.

Form



The input fields have the names ebText and ebNumber, and the button is named btSend.

The ADS-OCX must also be dragged on to the form. You will find more precise information in the documentation for the ADS-OCX.

Visual Basic program

```
Option Explicit

Private Sub Form_Load()
    ' port of the TwinCAT SMS server
    AdsOCX1.AdsAmsServerPort = 10400
End Sub

Private Sub btSend_Click()
    On Error GoTo ERRORHANDLER
    Dim message As String

    ' create the XML message
    message = "<Msg><Targets><Target><No>" & _
    ebNumber & _
    "</No></Target></Targets><Body><![CDATA[" & _
    ebText & _
    "]]></Body></Msg>"

    AdsOCX1.AdsSyncWriteStringReq &H4000, 1, LenB(message), message

    Exit Sub
ERRORHANDLER:
    MsgBox "Error " & Err.Number & ": " & Err.Description
End Sub
```

3.2.3.2 SMS Function Block in ST

Source text: https://infosys.beckhoff.com/content/1033/tcsmssmtpsrvce/Resources/11386391819/.exe

Task

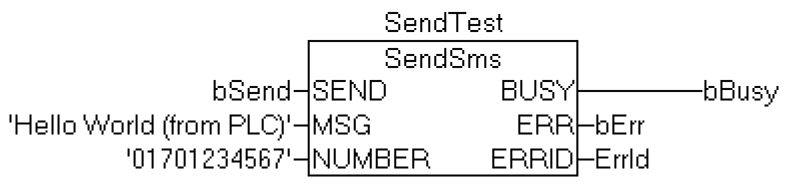
Simple function block for sending an SMS.

Description

The message and the telephone number to be called are provided via inputs.

The rising edge of the SEND input initiates transmission.

Calling the SMS Function Block



Implementation

The PlcSystem.lib library must be linked for the ADS block.

Declaration of the Variables

```
FUNCTION_BLOCK SendSms
VAR_INPUT
        SEND: BOOL;
        MSG: STRING(80);
        NUMBER: STRING;
END_VAR
VAR_OUTPUT
        BUSY: BOOL;
        ERR: BOOL;
        ERRID: UDINT;
END_VAR
VAR
        AdsWr: ADSWRITE;
        XmlMsg: STRING(250);
END_VAR
```

ST Code of the Function Block

```
XmlMsg :=       CONCAT('<Msg><Targets><Target><No>',
                CONCAT(NUMBER,
                CONCAT('</No></Target></Targets><Body><![CDATA[',
                CONCAT(MSG,
                         ']]></Body></Msg>'))));
AdsWr.PORT := 10400;
AdsWr.NETID := ''; (* local Net ID*)
AdsWr.IDXGRP := 16#4000;
AdsWr.IDXOFFS := 1;
AdsWr.LEN := SIZEOF(XmlMsg);
AdsWr.SRCADDR := ADR(XmlMsg);
AdsWr.TMOUT := T#5s;
AdsWr.WRITE := SEND;

AdsWr(); (* call the ADS function block *)

BUSY := AdsWr.BUSY;
ERR := AdsWr.ERR;
ERRID := AdsWr.ERRID;
```

3.3 Fault Finding

There are a number of reasons why an SMS may fail to be sent with the SendSMS function block or SMS COM Server:

- no connection to the GSM modem
- incorrectly configured communication settings of SMS COM Server
- incorrect call to the ADS service
- the use of an unsupported GSM modem
- incorrectly configured serial terminal (Advanced or Standard, 3 byte / 5 byte, speed, ...)
- incorrect telephone number
- PIN required (the SIM card must not be protected by a PIN)
- Serial terminal not initialised (call KL6Init)
- incorrect in GSM network

A variety of tools are available to look for these errors:

Using the Log File

Keeping records in a log file can be activated with the TwinCAT SMS Server Configurator. Once this has been done, all the messages sent, and the errors are written into the TcSmsSrvCfg.xml file. The file can be found in the TwinCAT installation directory.

NT Event Log

Errors when sending messages are also always recorded in the NT Event Log. The Event Log can be opened through the TwinCAT icon on the task bar.

ADS Error Messages

If the call to an ADS Function fails, the error is coded in the function's return value. A list of these error codes can be found under ADS Return Codes.

Configuration of the Terminal

The serial terminal can be configured in different ways. Terminals that have been differently configured, have to some extent a different representation in the process image (3 byte /5 byte terminals, advanced/standard). It must be noted that the ComLib library must be appropriate for the terminal configuration. See also the documentation for the KL6xxx and the ComLib documentation:

It is also important that the terminal transmission speed be matched to that of the modem in use.

Sending a Test SMS

A test SMS can easily be sent with the Visual Basic example program, to find out whether an error lies with the ADS call or in the configuration of the SMS Server.

Sending a Test SMS using a Mobile Telephone

To find out whether the SIM card is correctly configured, it can be inserted into an ordinary mobile phone and used to send an SMS. It should not be necessary to enter a PIN number here.

Network Selection with the Westermo GS-01

GS-01 has several variations for the various networks in Europe and in the USA. The lamp on the front of the modem indicates whether a network is available. The lamp flashes if the modem is connected to a network. If the lamp is continuously illuminated, the fault finding section should be consulted in the Westermo manual.

3.4 Cable for KL6001

A cable between the GSM modem and the KL6001 serial terminal should be wired as follows.

This cable uses hardware handshaking (RTS / CTS). It has been tested with the Westermo GS-01 and the Siemens S53i.

Connection	9 pin SUB-D socket	KL6001
RxD	2	5
TxD	3	1
GND	5	3
RTS	7	2
CTS	8	6

3.5 7 Bit GSM default alphabet coding

The TwinCAT SMS server encodes and transmits the SMS messages in accordance with the 7-bit standard alphabet. The special, non-printing ASCII characters (0x00..0x31) are not, however, automatically converted into the corresponding 7-bit code. In order to transmit these characters, the SMS string must first be appropriately formatted in accordance with the table below.

Example:

An SMS with the following text is to be sent from the PLC:

'Total: 100.89€, SmsSrv@Beckhoff.com'

The PLC string must have the following format:

'Total: 100.89$1B$65, SmsSrv$80Beckhoff.com'

7 bit default GSM alphabet as specified by GSM 03.38.		**8 bit ANSI alphabet**		
Character	**Character name**	**Hex**	**Dec**	**TwinCAT PLC string constant**
@	COMMERCIAL AT	0x00	0	$80 or every other number >= 0x80H
£	POUND SIGN	0x01	1	$01
$	DOLLAR SIGN	0x02	2	$02
¥	YEN SIGN	0x03	3	$03
è	LATIN SMALL LETTER E WITH GRAVE	0x04	4	$04
é	LATIN SMALL LETTER E WITH ACUTE	0x05	5	$05
ù	LATIN SMALL LETTER U WITH GRAVE	0x06	6	$06
ì	LATIN SMALL LETTER I WITH GRAVE	0x07	7	$07
ò	LATIN SMALL LETTER O WITH GRAVE	0x08	8	$08
Ç	LATIN CAPITAL LETTER C WITH CEDILLA	0x09	9	$09
	LINE FEED	0x0A	10	$0A or $N
Ø	LATIN CAPITAL LETTER O WITH STROKE	0x0B	11	$0B
ø	LATIN SMALL LETTER O WITH STROKE	0x0C	12	$0C
	CARRIAGE RETURN	0x0D	13	$0D or $R
Å	LATIN CAPITAL LETTER A WITH RING ABOVE	0x0E	14	$0E

7 bit default GSM alphabet as specified by GSM 03.38.		8 bit ANSI alphabet		
Character	**Character name**	**Hex**	**Dec**	**TwinCAT PLC string constant**
å	LATIN SMALL LETTER A WITH RING ABOVE	0x0F	15	$0F
?	GREEK CAPITAL LETTER DELTA	0x10	16	$10
_	LOW LINE	0x11	17	$11
F	GREEK CAPITAL LETTER PHI	0x12	18	$12
G	GREEK CAPITAL LETTER GAMMA	0x13	19	$13
?	GREEK CAPITAL LETTER LAMBDA	0x14	20	$14
O	GREEK CAPITAL LETTER OMEGA	0x15	21	$15
?	GREEK CAPITAL LETTER PI	0x16	22	$16
?	GREEK CAPITAL LETTER PSI	0x17	23	$17
S	GREEK CAPITAL LETTER SIGMA	0x18	24	$18
T	GREEK CAPITAL LETTER THETA	0x19	25	$19
?	GREEK CAPITAL LETTER XI	0x1A	26	$1A
	ESCAPE TO EXTENSION TABLE	0x1B	27	$1B
	FORM FEED	0x1B 0x0A	27 10	$1B$0A
^	CIRCUMFLEX ACCENT	0x1B 0x14	27 20	$1B$14
{	LEFT CURLY BRACKET	0x1B 0x28	27 40	$1B$28
}	RIGHT CURLY BRACKET	0x1B 0x29	27 41	$1B$29
\	REVERSE SOLIDUS (BACKSLASH)	0x1B 0x2F	27 47	$1B$2F
[	LEFT SQUARE BRACKET	0x1B 0x3C	27 60	$1B$3C
~	TILDE	0x1B 0x3D	27 61	$1B$3D
]	RIGHT SQUARE BRACKET	0x1B 0x3E	27 62	$1B$3E
|	VERTICAL BAR	0x1B 0x40	27 64	$1B$40
€	EURO SIGN	0x1B 0x65	27 101	$1B$65
Æ	LATIN CAPITAL LETTER AE	0x1C	28	$1C
æ	LATIN SMALL LETTER AE	0x1D	29	$1D
ß	LATIN SMALL LETTER SHARP S (German)	0x1E	30	$1E

7 bit default GSM alphabet as specified by GSM 03.38.		8 bit ANSI alphabet		
Character	**Character name**	**Hex**	**Dec**	**TwinCAT PLC string constant**
É	LATIN CAPITAL LETTER E WITH ACUTE	0x1F	31	$1F
	SPACE	0x20	32	$20 or ' '
!	EXCLAMATION MARK	0x21	33	!
"	QUOTATION MARK	0x22	34	"
#	NUMBER SIGN	0x23	35	#
¤	CURRENCY SIGN	0x24	36	$24 or $$
%	PERCENT SIGN	0x25	37	%
&	AMPERSAND	0x26	38	&
'	APOSTROPHE	0x27	39	$27 or $'
(	LEFT PARENTHESIS	0x28	40	(
)	RIGHT PARENTHESIS	0x29	41	)
*	ASTERISK	0x2A	42	*
+	PLUS SIGN	0x2B	43	+
,	COMMA	0x2C	44	,
-	HYPHEN-MINUS	0x2D	45	-
.	FULL STOP	0x2E	46	.
/	SOLIDUS (SLASH)	0x2F	47	/
0	DIGIT ZERO	0x30	48	0
1	DIGIT ONE	0x31	49	1
2	DIGIT TWO	0x32	50	2
3	DIGIT THREE	0x33	51	3
4	DIGIT FOUR	0x34	52	4
5	DIGIT FIVE	0x35	53	5
6	DIGIT SIX	0x36	54	6
7	DIGIT SEVEN	0x37	55	7
8	DIGIT EIGHT	0x38	56	8
9	DIGIT NINE	0x39	57	9
:	COLON	0x3A	58	:
;	SEMICOLON	0x3B	59	;
<	LESS-THAN SIGN	0x3C	60	<
=	EQUALS SIGN	0x3D	61	=
>	GREATER-THAN SIGN	0x3E	62	>
?	QUESTION MARK	0x3F	63	?
¡	INVERTED EXCLAMATION MARK	0x40	64	$40
A	LATIN CAPITAL LETTER A	0x41	65	A
B	LATIN CAPITAL LETTER B	0x42	66	B

7 bit default GSM alphabet as specified by GSM 03.38.		8 bit ANSI alphabet		
Character	**Character name**	**Hex**	**Dec**	**TwinCAT PLC string constant**
C	LATIN CAPITAL LETTER C	0x43	67	C
D	LATIN CAPITAL LETTER D	0x44	68	D
E	LATIN CAPITAL LETTER E	0x45	69	E
F	LATIN CAPITAL LETTER F	0x46	70	F
G	LATIN CAPITAL LETTER G	0x47	71	G
H	LATIN CAPITAL LETTER H	0x48	72	H
I	LATIN CAPITAL LETTER I	0x49	73	I
J	LATIN CAPITAL LETTER J	0x4A	74	J
K	LATIN CAPITAL LETTER K	0x4B	75	K
L	LATIN CAPITAL LETTER L	0x4C	76	L
M	LATIN CAPITAL LETTER M	0x4D	77	M
N	LATIN CAPITAL LETTER N	0x4E	78	N
O	LATIN CAPITAL LETTER O	0x4F	79	O
P	LATIN CAPITAL LETTER P	0x50	80	P
Q	LATIN CAPITAL LETTER Q	0x51	81	Q
R	LATIN CAPITAL LETTER R	0x52	82	R
S	LATIN CAPITAL LETTER S	0x53	83	S
T	LATIN CAPITAL LETTER T	0x54	84	T
U	LATIN CAPITAL LETTER U	0x55	85	U
V	LATIN CAPITAL LETTER V	0x56	86	V
W	LATIN CAPITAL LETTER W	0x57	87	W
X	LATIN CAPITAL LETTER X	0x58	88	X
Y	LATIN CAPITAL LETTER Y	0x59	89	Y
Z	LATIN CAPITAL LETTER Z	0x5A	90	Z
Ä	LATIN CAPITAL LETTER A WITH DIAERESIS	0x5B	91	$5B

7 bit default GSM alphabet as specified by GSM 03.38.		8 bit ANSI alphabet		
Character	**Character name**	**Hex**	**Dec**	**TwinCAT PLC string constant**
Ö	LATIN CAPITAL LETTER O WITH DIAERESIS	0x5C	92	$5C
Ñ	LATIN CAPITAL LETTER N WITH TILDE	0x5D	93	$5D
Ü	LATIN CAPITAL LETTER U WITH DIAERESIS	0x5E	94	$5E
§	SECTION SIGN	0x5F	95	$5F
¿	INVERTED QUESTION MARK	0x60	96	$60
a	LATIN SMALL LETTER A	0x61	97	a
b	LATIN SMALL LETTER B	0x62	98	b
c	LATIN SMALL LETTER C	0x63	99	c
d	LATIN SMALL LETTER D	0x64	100	d
e	LATIN SMALL LETTER E	0x65	101	e
f	LATIN SMALL LETTER F	0x66	102	f
g	LATIN SMALL LETTER G	0x67	103	g
h	LATIN SMALL LETTER H	0x68	104	h
i	LATIN SMALL LETTER I	0x69	105	i
j	LATIN SMALL LETTER J	0x6A	106	j
k	LATIN SMALL LETTER K	0x6B	107	k
l	LATIN SMALL LETTER L	0x6C	108	l
m	LATIN SMALL LETTER M	0x6D	109	m
n	LATIN SMALL LETTER N	0x6E	110	n
o	LATIN SMALL LETTER O	0x6F	111	o
p	LATIN SMALL LETTER P	0x70	112	p
q	LATIN SMALL LETTER Q	0x71	113	q
r	LATIN SMALL LETTER R	0x72	114	r
s	LATIN SMALL LETTER S	0x73	115	s
t	LATIN SMALL LETTER T	0x74	116	t

7 bit default GSM alphabet as specified by GSM 03.38.		8 bit ANSI alphabet		
Character	**Character name**	**Hex**	**Dec**	**TwinCAT PLC string constant**
u	LATIN SMALL LETTER U	0x75	117	u
v	LATIN SMALL LETTER V	0x76	118	v
w	LATIN SMALL LETTER W	0x77	119	w
x	LATIN SMALL LETTER X	0x78	120	x
y	LATIN SMALL LETTER Y	0x79	121	y
z	LATIN SMALL LETTER Z	0x7A	122	z
ä	LATIN SMALL LETTER A WITH DIAERESIS	0x7B	123	$7B
ö	LATIN SMALL LETTER O WITH DIAERESIS	0x7C	124	$7C
ñ	LATIN SMALL LETTER N WITH TILDE	0x7D	125	$7D
ü	LATIN SMALL LETTER U WITH DIAERESIS	0x7E	126	$7E
à	LATIN SMALL LETTER A WITH GRAVE	0x7F	127	$7F

3.6 Device control string syntax

The device-control string uses the syntax of the **mode** command. The string must have the same form as the **mode** command's command-line arguments. For further information on **mode** command syntax, refer to the end-user documentation for your operating system.

Syntax

```
modecomm[:] [baud=b] [parity=p] [data=d] [stop=s] [to={on|off}] [xon={on|off}] [odsr={on|off}]
[octs={on|off}] [dtr={on|off|hs}] [rts={on|off|hs|tg}] [idsr={on|off}]
```

Parameters

com *m*[:]	Specifies the number of the port for asynchronous communications (COM).
baud= *b*	Specifies the transmission rate in bits per second. The following table lists valid abbreviations for *b* and its related baud rate.
parity= *p*	Specifies how the system uses the parity bit to check for transmission errors. The following table lists the valid values: *p*. The default value is **e**. Not all computers support the values **m** and **s**.
data= *d*	Specifies the number of data bits in a character. Valid values for **d** are in the range 5 to 8. The default value is 7. Not all computers support the values 5 and 6.

stop= ***s***	Specifies the number of stop bits that define the end of a character: 1, 1.5 or 2. If the baud rate is 110, the default value is 2. Otherwise, the default value is 1. Not all computers support the value 1.5.
to={on|off}	Specifies whether the endless timeout processing is on or off. The default value is off.
xon={on|off}	Specifies whether the xon or xoff protocol for data-flow control is on or off.
odsr={on|off}	Specifies whether output handshaking that uses the Data Set Ready (DSR) signal is on or off.
octs={on|off}	Specifies whether output handshaking that uses the Clear To Send (CTS) signal is on or off.
dtr={on|off|hs}	Specifies whether the Data Terminal Ready (DTR) signal is on or off, or set to handshake.
rts={on|off|hs|tg}	Specifies whether the Request To Send (RTS) signal is set to on, off, handshake or toggle.
idsr={on|off}	Specifies whether the DSR signal is used or not.

Things to know:

- For a string such as **96,n,8,1** or any other older-form **mode** string that doesn't end with an **x** or a **p**:

  fInX, **fOutX**, **fOutXDsrFlow** and **fOutXCtsFlow** are all set to FALSE

  fDtrControl is set to DTR_CONTROL_ENABLE

  fRtsControl is set to RTS_CONTROL_ENABLE
- For a string such as **96,n,8,1,x** or any other older-form **mode** string that ends with an **x**:

  fInX and **fOutX** are both set to TRUE

  fOutXDsrFlow and **fOutXCtsFlow** are both set to FALSE

  fDtrControl is set to DTR_CONTROL_ENABLE

  fRtsControl is set to RTS_CONTROL_ENABLE
- For a string such as **96,n,8,1,p** or any other older-form **mode** string that ends with a **p**:

  fInX and **fOutX** are both set to FALSE

  fOutXDsrFlow and **fOutXCtsFlow** are both set to TRUE

  fDtrControl is set to DTR_CONTROL_HANDSHAKE

  fRtsControl is set to RTS_CONTROL_HANDSHAKE

4 SMTP Server

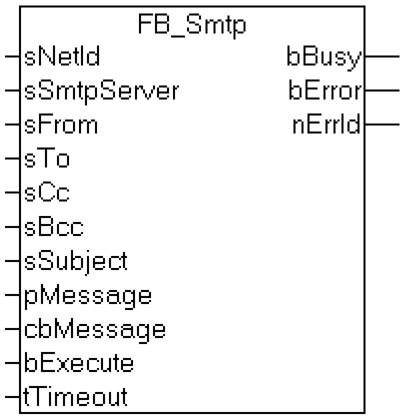
With the TwinCAT SMTP Server it is possible to send eMails directly out of the PLC. After a successful installation the server will be started together with TwinCAT. The server will be addressed over ADS out of the PLC. There are several function blocks available in the PLC to send eMails:

- FB_Smtp [▸ 35]
- FB_SmtpV2 [▸ 41]
- FB_SmtpAttach [▸ 37]
- FB_SmtpFull [▸ 39]
- FB_SmtpV3 [▸ 43]
- FB_SmtpV3_Full [▸ 45]

Encryption:

Since TC SMTP server version 1.0.14 SSL/STARTTLS encryption is supported.

4.1 FB_Smtp



The block sends a byte stream to a remote ADS device via ADS. The TwinCAT ADS Smtp service must be running on the remote ADS device, so that the byte stream can be received and processed into an e-mail. Once the byte stream has been processed the e-mail is sent.
Note that password checking must be disabled on the SMTP server, since the TwinCAT ADS Smtp service does not register on the server via password checking.

VAR_INPUT

```
VAR_INPUT
    sNetId          : T_AmsNetID;    (* AmsNetID *)
    sSmtpServer     : T_MaxString;   (* Smtp-Server address (IP or Name) *)
    sFrom           : T_MaxString;   (* Sender string *)
    sTo             : T_MaxString;   (* To recipient string *)
    sCc             : T_MaxString;   (* Cc recipient string *)
    sBcc            : T_MaxString;   (* Bcc recipient string *)
    sSubject        : T_MaxString;   (* Subject string *)
    pMessage        : DWORD;        (* Pointer to the message *)
    cbMessage       : UDINT;        (* Messagelenght to send *)
    bExecute        : BOOL;
    tTimeout        : TIME := T#20s;
END_VAR
```

sNetId: AmsNetID on which the TwinCAT SMS server runs.

sSmtpServer: Name or IP of the Smtp server.

sFrom: A string containing the e-mail address of the sender. A sender must be specified. The string is limited to 255 characters.

sTo: A string containing the e-mail address of the recipient. Several addresses can be specified, separated by semicolon. At least one recipient has to be specified. The string is limited to 255 characters.

sCc: A string containing an e-mail address of a further recipient (cc=carbon copy). This string can also be empty. A copy of the e-mail is sent to this recipient. The e-mail address of this recipient is **visible** to other recipients. It is possible to enter multiple recipient addresses separated by semicolons. The string is limited to 255 characters.

sBcc: A string containing the e-mail address of a further recipient (Bcc = blind carbon copy). This string can also be empty. A copy of the e-mail is sent to this\these recipient\s. The e-mail address of this recipient is not visible to other recipients. It is possible to enter multiple recipient addresses separated by semicolons. The string is limited to 255 characters.

sSubject: A string containing the subject line for the e-mail. The e-mail may be sent without subject, in which case the name of the sending computer is automatically entered in the subject line (e.g. "Mail sent from: CX_00762C"). The string for the subject line is limited to 255 characters.

pMessage: The address (a pointer) to a null-terminated string containing the e-mail text. The e-mail may be sent without body text, in which case the date and time are entered automatically (e.g. "Mail sent at: Thu, 23 Mar 2006 02:31:44 -0800"). The address of the string can be determined with the ADR operator.

cbMessage: Length of the e-mail text. The length can be determined through the LEN operator.

bExecute: The function block is activated by a rising edge at this input.

tTimeout: Maximum time allowed for the execution of the command.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy   : BOOL;
    bError  : BOOL;
    nErrId  : UDINT;
END_VAR
```

bBusy: this output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the tTimeOut input.

bError: this output is switched to TRUE if an error occurs during the execution of a command. The command-specific error code is contained in iErrorId.

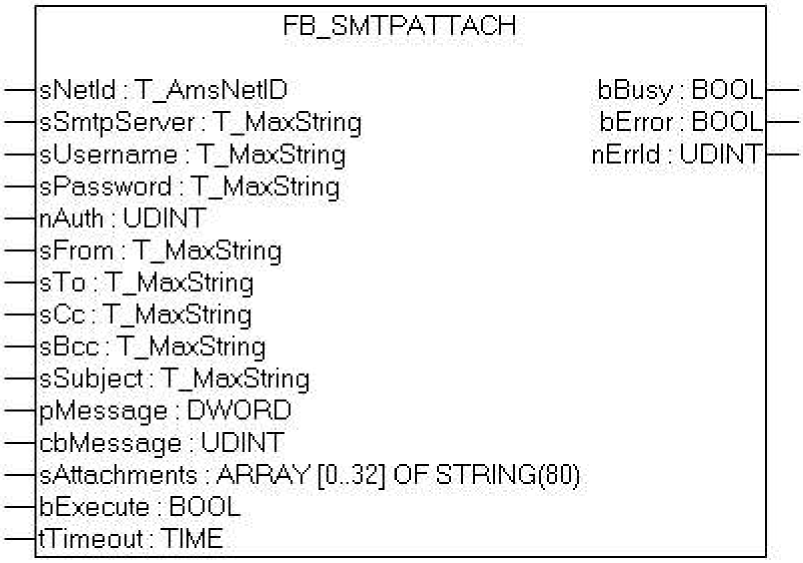
nErrId: contains the command-specific error code of the most recently executed command (see table [▸ 50]).

- Make sure that you have no \0 digits within the byte array, because the message will be aborted there.
- The maximum number of characters that may be used in a message is 510,725 - 1275 characters are available for From, To, Cc, Bcc and Subject.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8.0 and above		TcSmtp.lib

4.2 FB_SmtpAttach



The block sends a byte stream to a remote ADS device via ADS. The TwinCAT ADS Smtp service must be running on the remote ADS device, so that the byte stream can be received and processed into an e-mail. Once the byte stream has been processed the e-mail is sent.

VAR_INPUT

```
VAR_INPUT
 sNetId      : T_AmsNetID;        (* AmsNetID *)
 sSmtpServer : T_MaxString;       (* Smtp Server addres (IP or Name)*)
 sUsername   : T_MaxString;       (* Smtp Username *)
 sPassword   : T_MaxString;       (* Smtp Password *)
 nAuth       : UDINT;            (* Smtp Auth Type *)
 sFrom       : T_MaxString;       (* Sender stzring *)
 sTo         : T_MaxString;       (* To recipient string *)
 sCc         : T_MaxString;       (* Cc recipient string *)
 sBcc        : T_MaxString;       (* Bcc recipient string *)
 sSubject    : T_MaxString;       (* Subject string *)
 pMessage    : DWORD;            (* Pointer to the message *)
 cbMessage   : UDINT;            (* Messagelenght in byte to send *)
 sAttachments: ARRAY [0..32] OFSTRING;
 bExecute    : BOOL;
 tTimeout    : TIME := T#20s;
END_VAR
```

sNetId: AmsNetID on which the TwinCAT SMS server runs.

sSmtpServer: Name or IP of the Smtp server.

sUsername: Username for the Smtp Server.

sPassword: Password for the Smtp Server.

nAuth: Smtp Auth Type:
0 = AUTH NONE
1 = RESERVED
2 = AUTH LOGIN
3 = AUTH NTLM
4 = AUTH PLAIN

sFrom: A string containing the e-mail address of the sender. A sender must be specified. The string is limited to 255 characters.

sTo: A string containing the e-mail address of the recipient. Several addresses can be specified, separated by semicolon. At least one recipient has to be specified. The string is limited to 255 characters.

sCc: A string containing an e-mail address of a further recipient (cc=carbon copy). This string can also be empty. A copy of the e-mail is sent to this recipient. The e-mail address of this recipient is **visible** to other recipients. It is possible to enter multiple recipient addresses separated by semicolons. The string is limited to 255 characters.

sBcc: A string containing the e-mail address of a further recipient (Bcc = blind carbon copy). This string can also be empty. A copy of the e-mail is sent to this\these recipient\s. The e-mail address of this recipient is not visible to other recipients. It is possible to enter multiple recipient addresses separated by semicolons. The string is limited to 255 characters.

sSubject: A string containing the subject line for the e-mail. The e-mail may be sent without subject, in which case the name of the sending computer is automatically entered in the subject line (e.g. "Mail sent from: CX_00762C"). The string for the subject line is limited to 255 characters.

pMessage: The address (a pointer) to a null-terminated string containing the e-mail text. The e-mail may be sent without body text, in which case the date and time are entered automatically (e.g. "Mail sent at: Thu, 23 Mar 2006 02:31:44 -0800"). The address of the string can be determined with the ADR operator.

cbMessage: Length of the e-mail text. The length can be determined through the LEN operator.

bExecute: The function block is activated by a rising edge at this input.

sAttachments: Array containing filenames

tTimeout: Maximum time allowed for the execution of the command.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy   : BOOL;
    bError  : BOOL;
    nErrId  : UDINT;
END_VAR
```

bBusy: the output variable remains TRUE until the function block has executed a command, but only until tTimeOut has expired.

bError : the output variable is switched to TRUE as soon as an error occurs during the execution of the command. The command-specific error is contained in iErrorId.

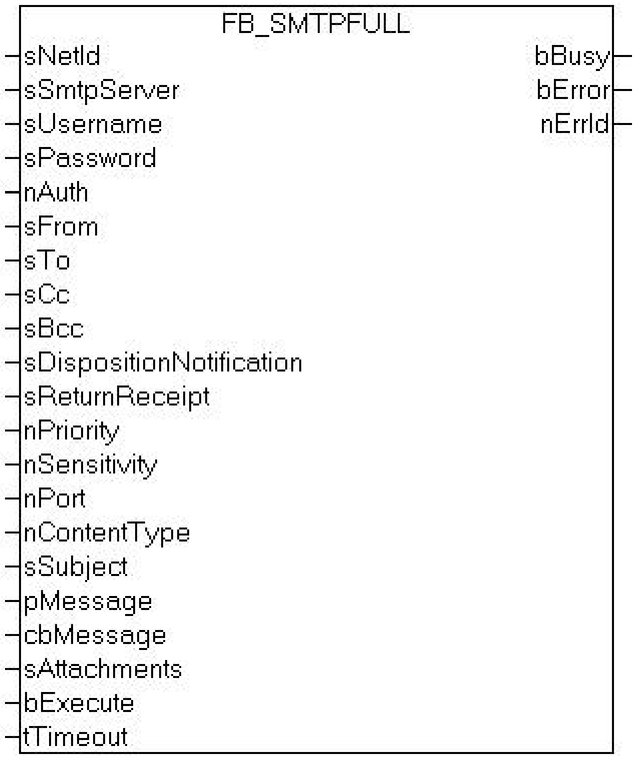
nErrId: contains the command-specific error code of the most recently executed command (see table [▸ 50]).

- Make sure that you do not use \o within the byte-arrays, otherwise the message will be stopped.
- The maximum number of characters in a message is 510,725 - in total you have 1275 characters for From, To, Cc, Bcc and Subject.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8.0 and above		TcSmtp.lib

4.3 FB_SmtpFull



This function block communicates over ADS with the TwinCAT SMTP Server. It offers a wide range of mail functionalities as for example the prioritization of emails out of the PLC. The individual parameters will be described in detail in this documentation.

VAR_INPUT

```
VAR_INPUT
    sNetId                  : T_AmsNetID;       (* AmsNetID *)
    sSmtpServer             : T_MaxString;      (* Smtp Server addres ( IP or Name) *)
    sUsername               : T_MaxString;      (* Smtp Username *)
    sPassword               : T_MaxString;      (* Smtp Password *)
    nAuth                   : UDINT;         (* Smtp Auth Type*)
    sFrom                   : T_MaxString;      (* Sender stzring *)
    sTo                     : T_MaxString;      (* To recipient string *)
    sCc                     : T_MaxString;      (* Cc recipient string *)
    sBcc                    : T_MaxString;      (* Bcc recipient string *)
    sDispositionNotification: T_MaxString;       (* Disposition notification recipent string *)
    sReturnReceipt          : T_MaxString;      (* Return recipent string *)
    nPriority               : UDINT;           (* Priority value *)
    nSensitivity            : UDINT;         (* Sensitivity value *)
    nPort                   : UDINT;         (* Communication port *)
    nContentType            : UDINT;         (* Content type *)
    sSubject                : T_MaxString;      (* Subject string *)
    pMessage                : DWORD;           (* Pointer to the message *)
    cbMessage               : UDINT;           (* Messagelenght in byte to send *)
    sAttachments            : ARRAY [0..32] OF STRING;    (* Different attachments *)
    bExecute                : BOOL;            (* Trigger flag *)
    tTimeout                : TIME := T#20s;          (* Communication timeout *)
END_VAR
```

sNetId: AmsNetID on which the TwinCAT SMTP server runs.

sSmtpServer: Name or IP of the SMTP server.

sUsername: Username for the SMTP server.

sPassword: Password for the SMTP server.

nAuth: Smtp Auth Type:
0 = AUTH NONE
1 = RESERVED
2 = AUTH LOGIN
3 = AUTH NTLM
4 = AUTH PLAIN

sFrom: A string containing the email address of the sender. A sender must be specified. The string is limited to 255 characters.

sTo: A string containing the email address of the recipient. Several addresses can be specified, separated by semicolon. At least one recipient has to be specified. The string is limited to 255 characters.

sCc: A string containing an email address of a further recipient (cc=carbon copy). This string can also be empty. A copy of the email is sent to this recipient. The email address of this recipient is **visible** to other recipients. It is possible to enter multiple recipient addresses separated by semicolons. The string is limited to 255 characters.

sBcc: A string containing the email address of a further recipient (Bcc = blind carbon copy). This string can also be empty. A copy of the email is sent to this\these recipient\s. The email address of this recipient is not visible to other recipients. It is possible to enter multiple recipient addresses separated by semicolons. The string is limited to 255 characters.

sDispositionNotification: The mail address which is given to this parameter receives an return receipt of the recipients under sTo and sCc. The condition precedent is that the return receipt will be send by the recipients.

sReturnReceipt: An acknowledgment of transfer will be send to this mail address.

nPriority: With this parameter you can set the priority of the mail:
1 = Highest
2 = not used
3 = Normal
4 = not used
5 = Lowest

nSensitivity: With this parameter you can set the confidentiality of the message:
0 = Private
1 = Personal
2 = Normal
3 = Confidential

nPort: You can choose the communication-port here. If you do not enter an own port it will be accessed to the default-port 25.

nContentType: With this parameter it is possible to make a HTML-code which is given per pointer (pMessage) and size (cbMessage) to a string variable readable in the mail**.**

sSubject: A string containing the subject line for the e-mail. The email may be sent without subject, in which case the name of the sending computer is automatically entered in the subject line (e.g. "Mail sent from: CX_00762C"). The string for the subject line is limited to 255 characters.

pMessage: The address (a pointer) to a null-terminated string containing the email text. The email may be sent without body text, in which case the date and time are entered automatically (e.g. "Mail sent at: Thu, 23 Mar 2006 02:31:44 -0800"). The address of the string can be determined with the ADR operator.

cbMessage: Length of the email text. The length can be determined through the LEN operator.

bExecute: The function block is activated by a rising edge at this input.

sAttachments: Array of filenames

tTimeout: Maximum time allowed for the execution of the command.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy   : BOOL;
    bError  : BOOL;
    nErrId  : UDINT;
END_VAR
```

bBusy: the output remains TRUE until the function block has executed a command, but only until tTimeOut has expired.

bError : the output is switched to TRUE as soon as an error occurs during the execution of the command. The command-specific error is contained in iErrorId.

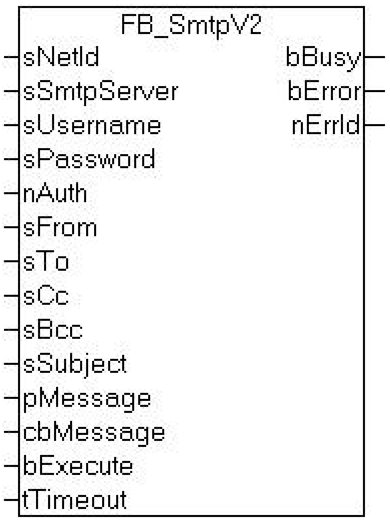
nErrId: contains the command-specific error code of the most recently executed command (see table [▶ 50]).

- Make sure that you do not use \o within the byte-arrays, otherwise the message will be stopped.
- The maximum number of characters in a message is 510,725 - in total you have 1275 characters for From, To, Cc, Bcc and Subject.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.10. and above		TcSmtp.lib

4.4 FB_SmtpV2



The block sends a byte stream to a remote ADS device via ADS. The TwinCAT ADS Smtp service must be running on the remote ADS device, so that the byte stream can be received and processed into an e-mail. Once the byte stream has been processed the e-mail is sent.

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetID;   (* AmsNetID *)
    sSmtpServer : T_MaxString;  (* Smtp Server addres ( IP or Name) *)
    sUsername   : T_MaxString;  (* Smtp Username *)
    sPassword   : T_MaxString;  (* Smtp Password *)
    nAuth       : UDINT;       (* Smtp Auth Type *)
    sFrom       : T_MaxString;  (* Sender stzring *)
    sTo         : T_MaxString;  (* To recipient string *)
    sCc         : T_MaxString;  (* Cc recipient string *)
    sBcc        : T_MaxString;  (* Bcc recipient string *)
    sSubject    : T_MaxString;  (* Subject string *)
    pMessage    : DWORD;       (* Pointer to the message *)
    cbMessage   : UDINT;       (* Messagelenght in byte to send *)
    bExecute    : BOOL;
    tTimeout    : TIME := T#20s;
END_VAR
```

sNetId: AmsNetID on which the TwinCAT Smtp server runs.

sSmtpServer: Name or IP of the Smtp server.

sUsername: Username for the Smtp Server.

sPassword: Password for the Smtp Server.

nAuth: Smtp Auth Type:
0 = AUTH NONE
1 = RESERVED
2 = AUTH LOGIN
3 = AUTH NTLM
4 = AUTH PLAIN

sFrom: A string containing the e-mail address of the sender. A sender must be specified. The string is limited to 255 characters.

sTo: A string containing the e-mail address of the recipient. Several addresses can be specified, separated by semicolon. At least one recipient has to be specified. The string is limited to 255 characters.

sCc: A string containing an e-mail address of a further recipient (cc=carbon copy). This string can also be empty. A copy of the e-mail is sent to this recipient. The e-mail address of this recipient is **visible** to other recipients. It is possible to enter multiple recipient addresses separated by semicolons. The string is limited to 255 characters.

sBcc: A string containing the e-mail address of a further recipient (Bcc = blind carbon copy). This string can also be empty. A copy of the e-mail is sent to this\these recipient\s. The e-mail address of this recipient is not visible to other recipients. It is possible to enter multiple recipient addresses separated by semicolons. The string is limited to 255 characters.

sSubject: A string containing the subject line for the e-mail. The e-mail may be sent without subject, in which case the name of the sending computer is automatically entered in the subject line (e.g. "Mail sent from: CX_00762C"). The string for the subject line is limited to 255 characters.

pMessage: The address (a pointer) to a null-terminated string containing the e-mail text. The e-mail may be sent without body text, in which case the date and time are entered automatically (e.g. "Mail sent at: Thu, 23 Mar 2006 02:31:44 -0800"). The address of the string can be determined with the ADR operator.

cbMessage: Length of the e-mail text. The length can be determined through the LEN operator.

bExecute: The function block is activated by a rising edge at this input.

tTimeout: Maximum time allowed for the execution of the command.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy   : BOOL;
    bError  : BOOL;
    nErrId  : UDINT;
END_VAR
```

bBusy: the output variable remains TRUE until the function block has executed a command, but only until tTimeOut has expired.

bError : the output variable is switched to TRUE as soon as an error occurs during the execution of the command. The command-specific error is contained in iErrorId.

nErrId: contains the command-specific error code of the most recently executed command (see table [▶ 50]).

- Make sure that you do not use \o within the byte-arrays, otherwise the message will be stopped.
- The maximum number of characters in a message is 510,725 - in total you have 1275 characters for From, To, Cc, Bcc and Subject.

Requirements

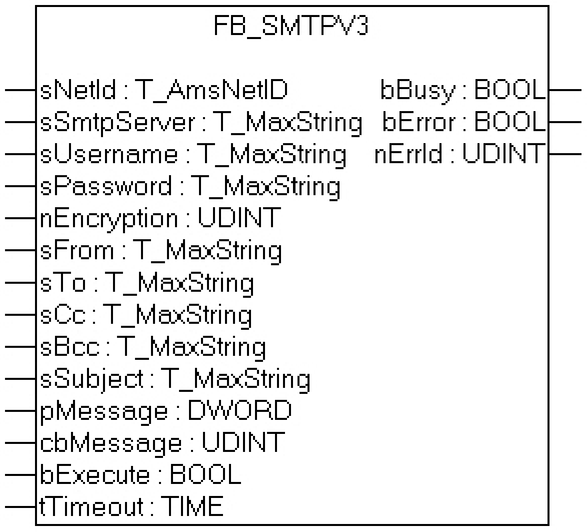
Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8.0 and above		TcSmtp.lib

Sample in ST:

```
PROGRAM MAIN
VAR
    FB_SmtpV2    : FB_SmtpV2;
    bSend        : BOOL;
    bBusy        : BOOL;
    bError       : BOOL;
    nErrID       : UDINT;
    sServer      : STRING := 'smtpserver';
    sMsg         : STRING := 'TcSmtpSrv is working properly';
    sSubject     : STRING :=  'TcSmtp Service Test';
    sUser        : STRING := 'username';
    sPassword    : STRING := 'password';
    sFrom        : STRING := 'emailfrom';
    sTo          : STRING := 'emailto';
    nAuth        : INT := 2;
END_VAR
```

```
FB_SmtpV2 (
    sNetId:= '',
    sSmtpServer:= sServer,
    sUsername:= sUser,
    sPassword:= sPassword,
    nAuth:= nAuth,
    sFrom:= sFrom,
    sTo:= sTo,
    sSubject:= sSubject,
    pMessage:= ADR(sMsg) ,
    cbMessage:= LEN(sMsg)+1,
    bExecute:= bSend,
    tTimeout:= t#20s,
    bBusy=> bBusy,
    bError=> bError,
    nErrId=> nErrID);
```

4.5 FB_SmtpV3



The block sends a byte stream to a remote ADS device via ADS. The TwinCAT ADS Smtp service must be running on the remote ADS device, so that the byte stream can be received and processed into an e-mail. Once the byte stream has been processed the e-mail is sent.

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetID;   (* AmsNetID *)
    sSmtpServer : T_MaxString;  (* Smtp Server addres ( IP or Name) *)
    sUsername   : T_MaxString;  (* Smtp Username *)
    sPassword   : T_MaxString;  (* Smtp Password *)
    nEncryption : UDINT;     (* 0=NONE, 1=STARTTLS, 2=SSL *)
    sFrom       : T_MaxString;  (* Sender stzring *)
    sTo         : T_MaxString;  (* To recipient string *)
    sCc         : T_MaxString;  (* Cc recipient string *)
    sBcc        : T_MaxString;  (* Bcc recipient string *)
    sSubject    : T_MaxString;  (* Subject string *)
    pMessage    : DWORD;     (* Pointer to the message *)
    cbMessage   : UDINT;     (* Messagelength in byte to send *)
    bExecute    : BOOL;
    tTimeout    : TIME := T#20s;
END_VAR
```

sNetId: AmsNetID on which the TwinCAT Smtp server runs.

sSmtpServer: Name or IP of the Smtp server.

sUsername: Username for the Smtp Server.

sPassword: Password for the Smtp Server.

nEncryption: Smtp encryption type:
0 = NONE
1 = STARTTLS
2 = SSL

sFrom: A string containing the e-mail address of the sender. A sender must be specified. The string is limited to 255 characters.

sTo: A string containing the e-mail address of the recipient. Several addresses can be specified, separated by semicolon. At least one recipient has to be specified. The string is limited to 255 characters.

sCc: A string containing an e-mail address of a further recipient (cc=carbon copy). This string can also be empty. A copy of the e-mail is sent to this recipient. The e-mail address of this recipient is **visible** to other recipients. It is possible to enter multiple recipient addresses separated by semicolons. The string is limited to 255 characters.

sBcc: A string containing the e-mail address of a further recipient (Bcc = blind carbon copy). This string can also be empty. A copy of the e-mail is sent to this\these recipient\s. The e-mail address of this recipient is not visible to other recipients. It is possible to enter multiple recipient addresses separated by semicolons. The string is limited to 255 characters.

sSubject: A string containing the subject line for the e-mail. The e-mail may be sent without subject, in which case the name of the sending computer is automatically entered in the subject line (e.g. "Mail sent from: CX_00762C"). The string for the subject line is limited to 255 characters.

pMessage: The address (a pointer) to a null-terminated string containing the e-mail text. The e-mail may be sent without body text, in which case the date and time are entered automatically (e.g. "Mail sent at: Thu, 23 Mar 2006 02:31:44 -0800"). The address of the string can be determined with the ADR operator.

cbMessage: Length of the e-mail text. The length can be determined through the LEN operator.

bExecute: The function block is activated by a rising edge at this input

tTimeout: Maximum time allowed for the execution of the command.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy   : BOOL;
    bError  : BOOL;
    nErrId  : UDINT;
END_VAR
```

bBusy: the output variable remains TRUE until the function block has executed a command, but only until tTimeOut has expired.

bError : the output variable is switched to TRUE as soon as an error occurs during the execution of the command. The command-specific error is contained in nErrId.

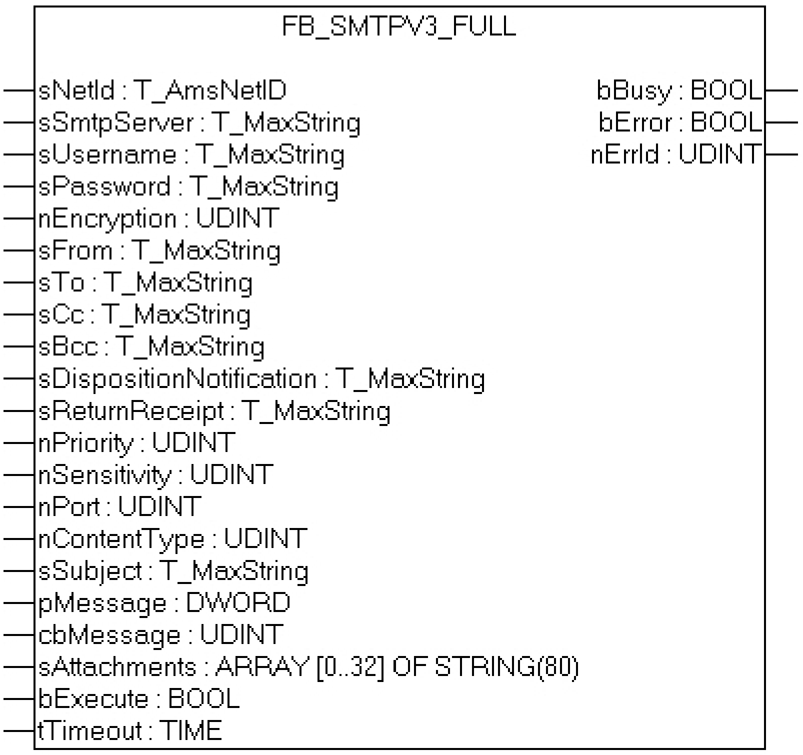
nErrId: contains the command-specific error code of the most recently executed command (see table [▸ 50]).

- Make sure that you do not use \o within the byte-arrays, otherwise the message will be stopped.
- The maximum number of characters in a message is 510,725 - in total you have 1275 characters for From, To, Cc, Bcc and Subject.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8.0 and above		TcSmtp.lib

4.6 FB_SmtpV3_Full



This function block communicates over ADS with the TwinCAT SMTP Server. It offers a wide range of mail functionalities as for example the prioritization of emails out of the PLC. The individual parameters will be described in detail in this documentation.

VAR_INPUT

```
VAR_INPUT
    sNetId                  : T_AmsNetID;       (* AmsNetID *)
    sSmtpServer             : T_MaxString;      (* Smtp Server addres ( IP or Name) *)
    sUsername               : T_MaxString;      (* Smtp Username *)
    sPassword               : T_MaxString;      (* Smtp Password *)
    nEncryption             : UDINT;        (* 0=NONE, 1=TLS, 2=SSL*)
    sFrom                   : T_MaxString;      (* Sender string *)
    sTo                     : T_MaxString;      (* To recipient string *)
    sCc                     : T_MaxString;      (* Cc recipient string *)
    sBcc                    : T_MaxString;      (* Bcc recipient string *)
    sDispositionNotification: T_MaxString;      (* Disposition notification recipent string *)
    sReturnReceipt          : T_MaxString;      (* Return recipent string *)
    nPriority               : UDINT;        (* Priority value *)
    nSensitivity            : UDINT;        (* Sensitivity value *)
    nPort                   : UDINT;        (* Communication port *)
    nContentType            : UDINT;        (* Content type *)
    sSubject                : T_MaxString;      (* Subject string *)
    pMessage                : DWORD;        (* Pointer to the message *)
    cbMessage               : UDINT;        (* Messagelength in byte to send *)
    sAttachments            : ARRAY [0..32] OF STRING;    (* Different attachments *)
    bExecute                : BOOL;         (* Trigger flag *)
    tTimeout                : TIME := T#20s;    (* Communication timeout *)
END_VAR
```

sNetId: AmsNetID on which the TwinCAT SMTP server runs.

sSmtpServer: Name or IP of the SMTP server.

sUsername: Username for the SMTP server.

sPassword: Password for the SMTP server.

nEncryption: Smtp encryption type:
0 = NONE
1 = STARTTLS
2 = SSL

sFrom: A string containing the email address of the sender. A sender must be specified. The string is limited to 255 characters.

sTo: A string containing the email address of the recipient. Several addresses can be specified, separated by semicolon. At least one recipient has to be specified. The string is limited to 255 characters.

sCc: A string containing an email address of a further recipient (cc=carbon copy). This string can also be empty. A copy of the email is sent to this recipient. The email address of this recipient is **visible** to other recipients. It is possible to enter multiple recipient addresses separated by semicolons. The string is limited to 255 characters.

sBcc: A string containing the email address of a further recipient (Bcc = blind carbon copy). This string can also be empty. A copy of the email is sent to this\these recipient\s. The email address of this recipient is not visible to other recipients. It is possible to enter multiple recipient addresses separated by semicolons. The string is limited to 255 characters.

sDispositionNotification: The mail address which is given to this parameter receives an return receipt of the recipients under sTo and sCc. The condition precedent is that the return receipt will be send by the recipients.

sReturnReceipt: An acknowledgment of transfer will be send to this mail address.

nPriority: With this parameter you can set the priority of the mail:
1 = Highest
2 = not used
3 = Normal
4 = not used
5 = Lowest

nSensitivity: With this parameter you can set the confidentiality of the message:
0 = Private
1 = Personal
2 = Normal
3 = Confidential

nPort: You can choose the communication-port here. If you do not enter an own port it will be accessed to the default-port 25.

nContentType: With this parameter it is possible to make a HTML-code which is given per pointer (pMessage) and size (cbMessage) to a string variable readable in the mail**.**

sSubject: A string containing the subject line for the e-mail. The email may be sent without subject, in which case the name of the sending computer is automatically entered in the subject line (e.g. "Mail sent from: CX_00762C"). The string for the subject line is limited to 255 characters.

pMessage: The address (a pointer) to a null-terminated string containing the email text. The email may be sent without body text, in which case the date and time are entered automatically (e.g., "Mail sent at: Thu, 23 Mar 2006 02:31:44 -0800"). The address of the string can be determined with the ADR operator.

cbMessage: Length of the email text. The length can be determined through the LEN operator.

bExecute: The function block is activated by a rising edge at this input.

sAttachments: Array of filenames

tTimeout: Maximum time allowed for the execution of the command.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy   : BOOL;
    bError  : BOOL;
    nErrId  : UDINT;
END_VAR
```

bBusy: the output remains TRUE until the function block has executed a command, but only until tTimeOut has expired.

bError : the output is switched to TRUE as soon as an error occurs during the execution of the command. The command-specific error is contained in iErrorId.

nErrId: contains the command-specific error code of the most recently executed command (see table [▶ 50]).

- Make sure that you do not use \o within the byte-arrays, otherwise the message will be stopped.
- The maximum number of characters in a message is 510,725 - in total you have 1275 characters for From, To, Cc, Bcc and Subject.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.10. and above		TcSmtp.lib

5 Samples

5.1 Sample: Sent mail via PLC

A rising edge at bStart causes a mail to be sent.

https://infosys.beckhoff.com/content/1033/tcsmssmtpsrvce/Resources/11386393227/.zip

The mail addresses and the SMTP server data must be adjusted beforehand.

Program-variables

```
PROGRAM MAIN
VAR
fbSendMail: FB_SmtpV3;
sMessage: STRING := 'Hello Beckhoff';
bStart: BOOL;
bBusy: BOOL;
bError: BOOL;
nErrId: UDINT;
nMails: UINT;
END_VAR
```

Program-code

```
fbSendMail(sNetId:= '',
sSmtpServer:= 'mail.company.com',
sUsername:= '',
sPassword:= '',
nEncryption:= 0,
sFrom:= 'machine@company.com',
sTo:= 'service@customer.com',
sSubject:= 'Mail sent via TwinCAT SMTP',
pMessage:= ADR(sMessage),
cbMessage:= SIZEOF(sMessage),
bExecute:= bStart,
bBusy=> bBusy,
bError=> bError,
nErrId=> nErrId);

IF NOT bError AND NOT bBusy AND bStart THEN
bStart := FALSE;
END_IF
```

Requirements

Development environment	Target system	PLC libraries to include
TwinCAT v2.10.0 or higher with (x86)	x86 or ARM	TcSmtp.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib will be included automatically)

5.2 Sample: Sending of HTML mails

With the FB_SmtpV3Full very extensive email functionalities are available for the PLC. Among other things, the email text is transferred in HTML code, which offers completely new formatting possibilities. This makes it very easy to transfer current measured values or similar in a structured form.

Download: https://infosys.beckhoff.com/content/1033/tcsmssmtpsrvce/Resources/11386394635/.zip

The mail addresses and the SMTP server data must be adjusted beforehand.

Program variables

```
PROGRAM MAIN
VAR
fbSmtpFull : FB_SmtpV3_Full;
sMessage_HTML    : STRING := '<!DOCTYPE html><html><body><p>Sent by TwinCAT SMTP.</p></body></html>';
bStart: BOOL;
bBusy: BOOL;
bError: BOOL;
nErrId: UDINT;
END_VAR
```

Program code

```
fbSmtpFull(
sNetId:= '',
sSmtpServer:= 'mail.company.com',
sUsername:= '',
sPassword:= '',
sFrom:= 'machine@company.com',
sTo:= 'service@customer.com',
nContentType:= 2,
sSubject:= 'Email from your Beckhoff PLC',
pMessage:= ADR(sMessage_HTML),
cbMessage:= SIZEOF(sMessage_HTML),
bExecute:= bStart,
bBusy=> bBusy,
bError=> bError,
nErrId=> nErrId);

IF NOT bError AND NOT bBusy AND bStart THEN
bStart := FALSE;
END_IF
```

Requirements

Requirements

Development environment	Target system	PLC libraries to include
TwinCAT v2.10.0 or higher with (x86)	x86 or ARM	TcSmtp.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib will be included automatically)

6 Annex

6.1 Error Codes

This list contain error codes of the TwinCAT supplement product SMTP Server. If you miss some error codes, please look under ADS return codes [▶ 20] or WinSockErrorCodes [▶ 51].

Hex	Dec	Description
< 0x8000	< 32778	ADS return code [▶ 20]
0x800A	32778	Not connected
0x800B	32779	Sender expected
0x800C	32780	Recipients expected
0x800D	32781	Send FROM command failed
0x800E	32782	Send DATA command failed
0x800F	32783	Send mail header failed
0x8010	32784	Send mail body failed
0x8011	32785	Send "end of mail indicator" failed
0x8012	32786	Send "RCPT" command failed
0x8013	32787	Server Response got no username request
0x8014	32788	Server Response got no password request
0x8015	32789	Unable to create socket connection
0x8016	32790	Authentication type not supported by smtp server
0x8017	32791	Wrong username or password
0x8018	32792	Not supported
0x8019	32793	Invalid hostname
0x801A	32794	Unable to send attachment
0x801B	32795	File not found
0x801C	32796	Invalid Version (New SMTP Server with old SMTP PLC library)
0x801D	32797	Unable to connect (Connection error => sometimes wrong port or wrong server)
0x801E	32798	Unable to create socket
0x801F	32799	WSA startup failed
0x8020	32800	Invalid hostname
0x8021	32801	Unecpcted response from server
0x8022	32802	Error while receiving data
0x8023	32803	No supported authentication methods found
0x8024	32804	Invalid parameter
0x80A0	32928	Security interface not found
0x80A1	32929	Unable to call security interface
0x80A2	32930	Security initialization failed
0x80A4	32932	Unable to create credentials
0x80A5	32933	SSL-handshake failed
0x80A6	32934	Invalid server credentials
0x80A7	32935	Unable to verify server
0x80A8	32936	Unable to encrypt message
0x80A9	32937	Unable to decrypt message

The following errors can occur in older versions of the server (< 1.0.14)

Hex	Dec	Description
0x000A	10	Not connected
0x000B	11	Sender expected
0x000C	12	Recipients expected
0x000D	13	Send FROM command failed
0x000E	14	Send DATA command failed
0x000F	15	Send mail header failed
0x0010	16	Send mail body failed
0x0011	17	Send "end of mail indicator" failed
0x0012	18	Send "RCPT" command failed
0x0064	100	General error
0x0065	101	Invalid parameter
0x0066	102	Funtion not loaded
0x0067	103	Dll not loaded
0x0068	104	TcSmtpDll.dll cannot load. Check the installation from the TcSmtpDll.dll.
0x80D3	211	System status, or system help reply
0x80D6	214	Help message [Information on how to use the receiver or the meaning of a particular non-standard command; this reply is useful only to the human user]
0x80FB	251	User not local; will forward to <forward-path>
0x8163	354	Start mail input; end with <CRLF>.<CRLF>
0x81A5	421	<domain> Service not available, closing transmission channel [This may be a reply to any command if the service knows it must shut down]
0x81C2	450	Requested mail action not taken: mailbox unavailable [E.g., mailbox busy]
0x81C3	451	Requested action aborted: error in processing
0x81C4	452	Requested action not taken: insufficient system storage
0x81F4	500	Syntax error, command unrecognized [This may include errors such as command line too long]
0x81F5	501	Syntax error in parameters or arguments.
0x81F6	502	Command not implemented.
0x81F7	503	Bad sequence of commands.
0x8504	504	Command parameter not implemented
0x8226	550	Requested action not taken: mailbox unavailable [E.g., mailbox not found, no access]
0x8227	551	User not local; please try <forward-path>
0x8228	552	Requested mail action aborted: exceeded storage allocation
0x8229	553	Requested action not taken: mailbox name not allowed [E.g., mailbox syntax incorrect]
0x8224	554	Transaction failed

6.2 Windows Socket Error Codes

The following table describes the possible error codes, returned by the WSAGetLastError function. The errors are sorted in alphabetical order. Some error codes that are defined in Winsock2.h are not returned. They are not included in the list.

Return Value	Description
WSAEINTR10004	Interrupted function call.blocking operation was interrupted by a call to WSACancelBlockingCall.
WSAEACCES 10013	Permission denied.An attempt was made to access a socket in a way forbidden by its access permissions. An example is using a broadcast address for sendto without broadcast permission being set using setsockopt(SO_BROADCAST). Another possible reason for the

Return Value	Description
	WSAEACCES error is that when the bind function is called (on Windows NT 4 SP4 or later), another application, service, or kernel mode driver is bound to the same address with exclusive access. Such exclusive access is a new feature of Windows NT 4 SP4 and later, and is implemented by using the SO_EXCLUSIVEADDRUSE option.
WSAEFAULT 10014	Bad address.The system detected an invalid pointer address in attempting to use a pointer argument of a call. This error occurs if an application passes an invalid pointer value, or if the length of the buffer is too small. For instance, if the length of an argument, which is a sockaddr structure, is smaller than the sizeof(sockaddr).
WSAEINVAL 10022	Invalid argument.Some invalid argument was supplied (for example, specifying an invalid level to the setsockopt function). In some instances, it also refers to the current state of the socket—for instance, calling accept on a socket that is not listening.
WSAEMFILE 10024	Too many open files.Too many open sockets. Each implementation may have a maximum number of socket handles available, either globally, per process, or per thread.
WSAEWOULDBLOCK 10035	Resource temporarily unavailable.This error is returned from operations on nonblocking sockets that cannot be completed immediately, for example recv when no data is queued to be read from the socket. It is a nonfatal error, and the operation should be retried later. It is normal for WSAEWOULDBLOCK to be reported as the result from calling connect on a nonblocking SOCK_STREAM socket, since some time must elapse for the connection to be established.
WSAEINPROGRESS 10036	Operation now in progress.A blocking operation is currently executing. Windows Sockets only allows a single blocking operation—per- task or thread—to be outstanding, and if any other function call is made (whether or not it references that or any other socket) the function fails with the WSAEINPROGRESS error.
WSAEALREADY 10037	Operation already in progress.An operation was attempted on a nonblocking socket with an operation already in progress—that is, calling connect a second time on a nonblocking socket that is already connecting, or canceling an asynchronous request (WSAAsyncGetXbyY) that has already been canceled or completed.
WSAENOTSOCK 10038	Socket operation on nonsocket.An operation was attempted on something that is not a socket. Either the socket handle parameter did not reference a valid socket, or for select, a member of an fd_set was not valid.
WSAEDESTADDRREQ 10039	Destination address required.A required address was omitted from an operation on a socket. For example, this error is returned if sendto is called with the remote address of ADDR_ANY.
WSAEMSGSIZE 10040	Message too long.A message sent on a datagram socket was larger than the internal message buffer or some other network limit, or the buffer used to receive a datagram was smaller than the datagram itself.
WSAEPROTOTYPE 10041	Protocol wrong type for socket.A protocol was specified in the socket function call that does not support the semantics of the socket type requested. For example, the ARPA Internet UDP protocol cannot be specified with a socket type of SOCK_STREAM.
WSAENOPROTOOPT 10042	Bad protocol option.An unknown, invalid or unsupported option or level was specified in a getsockopt or setsockopt call.
WSAEPROTONOSUPPORT 10043	Protocol not supported.The requested protocol has not been configured into the system, or no implementation for it exists. For example, a socket call requests a SOCK_DGRAM socket, but specifies a stream protocol.
WSAESOCKTNOSUPPORT 10044	Socket type not supported.The support for the specified socket type does not exist in this address family. For example, the optional type SOCK_RAW might be selected in a socket call, and the implementation does not support SOCK_RAW sockets at all.

Return Value	Description
WSAEOPNOTSUPP 10045	Operation not supported.The attempted operation is not supported for the type of object referenced. Usually this occurs when a socket descriptor to a socket that cannot support this operation is trying to accept a connection on a datagram socket.
WSAEPFNOSUPPORT 10046	Protocol family not supported.The protocol family has not been configured into the system or no implementation for it exists. This message has a slightly different meaning from WSAEAFNOSUPPORT. However, it is interchangeable in most cases, and all Windows Sockets functions that return one of these messages also specify WSAEAFNOSUPPORT.
WSAEAFNOSUPPORT 10047	Address family not supported by protocol family.An address incompatible with the requested protocol was used. All sockets are created with an associated address family (that is, AF_INET for Internet Protocols) and a generic protocol type (that is, SOCK_STREAM). This error is returned if an incorrect protocol is explicitly requested in the socket call, or if an address of the wrong family is used for a socket, for example, in sendto.
WSAEADDRINUSE 10048	Address already in use.Typically, only one usage of each socket address (protocol/IP address/port) is permitted. This error occurs if an application attempts to bind a socket to an IP address/port that has already been used for an existing socket, or a socket that was not closed properly, or one that is still in the process of closing. For server applications that need to bind multiple sockets to the same port number, consider using setsockopt (SO_REUSEADDR). Client applications usually need not call bind at all— connect chooses an unused port automatically. When bind is called with a wildcard address (involving ADDR_ANY), a WSAEADDRINUSE error could be delayed until the specific address is committed. This could happen with a call to another function later, including connect, listen, WSAConnect, or WSAJoinLeaf.
WSAEADDRNOTAVAIL 10049	Cannot assign requested address.The requested address is not valid in its context. This normally results from an attempt to bind to an address that is not valid for the local computer. This can also result from connect, sendto, WSAConnect, WSAJoinLeaf, or WSASendTo when the remote address or port is not valid for a remote computer (for example, address or port 0).
WSAENETDOWN 10050	Network is down.A socket operation encountered a dead network. This could indicate a serious failure of the network system (that is, the protocol stack that the Windows Sockets DLL runs over), the network interface, or the local network itself.
WSAENETUNREACH 10051	Network is unreachable.A socket operation was attempted to an unreachable network. This usually means the local software knows no route to reach the remote host.
WSAENETRESET 10052	Network dropped connection on reset.The connection has been broken due to keep-alive activity detecting a failure while the operation was in progress. It can also be returned by setsockopt if an attempt is made to set SO_KEEPALIVE on a connection that has already failed.
WSAECONNABORTED 10053	Software caused connection abort.An established connection was aborted by the software in your host computer, possibly due to a data transmission time-out or protocol error.
WSAECONNRESET 10054	Connection reset by peer.An existing connection was forcibly closed by the remote host. This normally results if the peer application on the remote host is suddenly stopped, the host is rebooted, the host or remote network interface is disabled, or the remote host uses a hard close (see setsockopt for more information on the SO_LINGER option on the remote socket). This error may also result if a connection was broken due to keep-alive activity detecting a failure while one or more operations are in progress. Operations that were in progress fail with WSAENETRESET. Subsequent operations fail with WSAECONNRESET.
WSAENOBUFS 10055	No buffer space available.An operation on a socket could not be performed because the system lacked sufficient buffer space or because a queue was full.

Return Value	Description
WSAEISCONN 10056	Socket is already connected.A connect request was made on an already-connected socket. Some implementations also return this error if sendto is called on a connected SOCK_DGRAM socket (for SOCK_STREAM sockets, the to parameter in sendto is ignored) although other implementations treat this as a legal occurrence.
WSAENOTCONN 10057	Socket is not connected.A request to send or receive data was disallowed because the socket is not connected and (when sending on a datagram socket using sendto) no address was supplied. Any other type of operation might also return this error—for example, setsockopt setting SO_KEEPALIVE if the connection has been reset.
WSAESHUTDOWN 10058	Cannot send after socket shutdown.A request to send or receive data was disallowed because the socket had already been shut down in that direction with a previous shutdown call. By calling shutdown a partial close of a socket is requested, which is a signal that sending or receiving, or both have been discontinued.
WSAETIMEDOUT 10060	Connection timed out.A connection attempt failed because the connected party did not properly respond after a period of time, or the established connection failed because the connected host has failed to respond.
WSAECONNREFUSED 10061	Connection refused.No connection could be made because the target computer actively refused it. This usually results from trying to connect to a service that is inactive on the foreign host—that is, one with no server application running.
WSAEHOSTDOWN 10064	Host is down.A socket operation failed because the destination host is down. A socket operation encountered a dead host. Networking activity on the local host has not been initiated. These conditions are more likely to be indicated by the error WSAETIMEDOUT.
WSAEHOSTUNREACH 10065	No route to host.A socket operation was attempted to an unreachable host. See WSAENETUNREACH.
WSAEPROCLIM 10067	Too many processes.A Windows Sockets implementation may have a limit on the number of applications that can use it simultaneously.WSAStartup may fail with this error if the limit has been reached.
WSASYSNOTREADY 10091	Network subsystem is unavailable.This error is returned by WSAStartup if the Windows Sockets implementation cannot function at this time because the underlying system it uses to provide network services is currently unavailable. Users should check: • That the appropriate Windows Sockets DLL file is in the current path. • That they are not trying to use more than one Windows Sockets implementation simultaneously. If there is more than one Winsock DLL on your system, be sure the first one in the path is appropriate for the network subsystem currently loaded. • The Windows Sockets implementation documentation to be sure all necessary components are currently installed and configured correctly.
WSAVERNOTSUPPORTED 10092	Winsock.dll version out of range.The current Windows Sockets implementation does not support the Windows Sockets specification version requested by the application. Check that no old Windows Sockets DLL files are being accessed.
WSANOTINITIALISED 10093	Successful WSAStartup not yet performed.Either the application has not called WSAStartup or WSAStartup failed. The application may be accessing a socket that the current active task does not own (that is, trying to share a socket between tasks), or WSACleanup has been called too many times.
WSAEDISCON 10101	Graceful shutdown in progress.Returned by WSARecv and WSARecvFrom to indicate that the remote party has initiated a graceful shutdown sequence.
WSATYPE_NOT_FOUND 10109	Class type not found.The specified class was not found.

Return Value	Description
WSAHOST_NOT_FOUND 11001	Host not found.No such host is known. The name is not an official host name or alias, or it cannot be found in the database(s) being queried. This error may also be returned for protocol and service queries, and means that the specified name could not be found in the relevant database.
WSATRY_AGAIN 11002	Nonauthoritative host not found.This is usually a temporary error during host name resolution and means that the local server did not receive a response from an authoritative server. A retry at some time later may be successful.
WSANO_RECOVERY 11003	This is a nonrecoverable error.This indicates that some sort of nonrecoverable error occurred during a database lookup. This may be because the database files (for example, BSD-compatible HOSTS, SERVICES, or PROTOCOLS files) could not be found, or a DNS request was returned by the server with a severe error.
WSANO_DATA 11004	Valid name, no data record of requested type.The requested name is valid and was found in the database, but it does not have the correct associated data being resolved for. The usual example for this is a host name-to-address translation attempt (using gethostbyname or WSAAsyncGetHostByName) which uses the DNS (Domain Name Server). An MX record is returned but no A record—indicating the host itself exists, but is not directly reachable.
WSA_INVALID_HANDLE OS dependent	Specified event object handle is invalid.An application attempts to use an event object, but the specified handle is not valid.
WSA_INVALID_PARAMETE R OS dependent	One or more parameters are invalid.An application used a Windows Sockets function which directly maps to a Windows function. The Windows function is indicating a problem with one or more parameters.
WSA_IO_INCOMPLETE OS dependent	Overlapped I/O event object not in signaled state.The application has tried to determine the status of an overlapped operation which is not yet completed. Applications that use WSAGetOverlappedResult (with the fWait flag set to FALSE) in a polling mode to determine when an overlapped operation has completed, get this error code until the operation is complete.
WSA_IO_PENDING OS dependent	Overlapped operations will complete later. The application has initiated an overlapped operation that cannot be completed immediately. A completion indication will be given later when the operation has been completed.
WSA_NOT_ENOUGH_MEM ORY OS dependent	Insufficient memory available.An application used a Windows Sockets function that directly maps to a Windows function. The Windows function is indicating a lack of required memory resources.
WSA_OPERATION_ABORT ED OS dependent	Overlapped operation aborted.An overlapped operation was canceled due to the closure of the socket, or the execution of the SIO_FLUSH command in WSAIoctl.
WSAINVALIDPROCTABLE OS dependent	Invalid procedure table from service provider.A service provider returned a bogus procedure table to Ws2_32.dll. (This is usually caused by one or more of the function pointers being null.)
WSAINVALIDPROVIDER OS dependent	Invalid service provider version number.A service provider returned a version number other than 2.0.
WSAPROVIDERFAILEDINIT OS dependent	Unable to initialize a service provider.Either a service provider's DLL could not be loaded (LoadLibrary failed) or the provider's WSPStartup/NSPStartup function failed.
WSASYSCALLFAILURE OS dependent	System call failure.Generic error code, returned under various conditions. Returned when a system call that should never fail does fail. For example, if a call to WaitForMultipleEvents fails or one of the registry functions fails trying to manipulate the protocol/namespace catalogs.Returned when a provider does not return SUCCESS and does not provide an extended error code. Can indicate a service provider implementation error.

More Information:
www.beckhoff.com/ts6350

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