Function block library

IT_Library_8

for PLCnext Engineer

Documentation for
PHOENIX CONTACT function blocks
PHOENIX CONTACT GmbH Co. KG
Flachsmarktstrasse 8
D-32825 Blomberg, Germany

This documentation is available in English only.

Table of Contents

1 Installation hint

If you did not specify a different directory during **library** installation all data in the MSI file will be unpacked to

c:\Users\Public\Documents\Phoenix Contact Libraries\PLCnext Engineer (former: PC Worx Engineer)

Please copy the library data to your PLCnext Engineer (former: PC Worx Engineer) working library directory.

If you did not specify a different directory during **PLCnext Engineer** installation the default PLCnext Engineer working library directory is

c:\Users\Public\Documents\PLCnext Engineer\Libraries (former: PC Worx Engineer\Libraries)

2 General information

This library offers function blocks for different IT applications: DHCP (Dynamic Host Configuration Protocol), DNS (Domain Name System), FTP (File Transfer Protocol), HTTP (Hypertext Transfer Protocol), SMTP (Simple Mail Transfer Protocol) and SNTP (Simple Network Time Protocol).

For using this library the IP_Com library is also required.

Note:

Always use the current IP_Com library available on the Phoenix Contact website. Integrate both the IT_Library and the IP_Com library into your project and make sure that the current IP_Com library is integrated into the IT_Library. If the IT_Library cannot be transferred, even though the current IP_Com library is integrated, check the path of the integrated library and change it if the path does not match the storage location of the current library.

3 Change notes

Library version	Library build	PLCnext Engineer version	Change notes	Supported PLCs
8	20200210	>= 2020.0 LTS	Released for 2020.0 LTS	AXC F 1152 (1151412) AXC F 2152 (2404267)
7	20191209	2019.0 LTS 2019.3 2019.6 2019.9 2020.0 LTS	Adapted to 2020.0 LTS	AXC F 2152 (2404267)
6	20191121	2019.0 LTS 2019.3 2019.6 2019.9	ITL_SMTP_Client: • Typo in wDiagCode fixed: 16#C110 -> 16#C101 • Bug fix in ITL_6_EXA_ITL_SMTP_Client.pcwex: udtSocket was not correctly connected.	AXC F 2152 (2404267)
6	20191029	2019.0 LTS 2019.3 2019.6 2019.9	ITL_SMTP_Client: Additional providers are supported.	AXC F 2152 (2404267)
5	20191010	2019.0 LTS 2019.3 2019.6 2019.9	Revised documentation	AXC F 2152 (2404267)
5	20191002	2019.0 LTS 2019.3 2019.6 2019.9	Adapted to 2019.9	AXC F 2152 (2404267)
4	20190924	2019.0 LTS 2019.3 2019.6	ITL_SNTP: Bug fix for “Stuck in state 997 in case of error”.	AXC F 2152 (2404267)
3	20190722	2019.0 LTS 2019.3 2019.6	Adapted to 2019.6	AXC F 2152 (2404267)
2	20190701	2019.0 LTS 2019.3	ITL_SNTP: Bug fix	AXC F 2152 (2404267)
1	20190630	2019.0 LTS 2019.3	Initial version.	AXC F 2152 (2404267)

New version number: Functional changes of at least one function block, incompatibilities (e.g. change of library format)

New build number: No functional changes, but changes in the MSI file (e.g. documentation update, additional examples)

Note: This library is only released for a cycle time of 20 ms or more.

4 Function blocks

Function block	Description	Version	Supported articles	License
ITL_DNS	This function block can be used to request the IP address assigned to a host name from a DNS server.	1	-	none
ITL_FTP_FileCopy	This block makes it possible to copy a file between FTP servers.	1	-	none
ITL_FTP_FileRW	This block allows writing to a file on an FTP server or reading from a file on an FTP server.	1	-	none
ITL_FTP_Mngt	Management function block for FTP-protocol.	1	-	none
ITL_SMTP_Client	This function block can be used to sent mails via SMTP.	3	-	none
ITL_SNTP_Client	The SNTP_Client block determines the current time of an (S)NTP server via the SNTP protocol and makes this available at its outputs for further processing.	3	-	none
ITL_SNTP_Diag_Info	In case of an error at the ITL_SNTP_Client, this block shows the diagnostics of the block as a text in German.	1	-	none

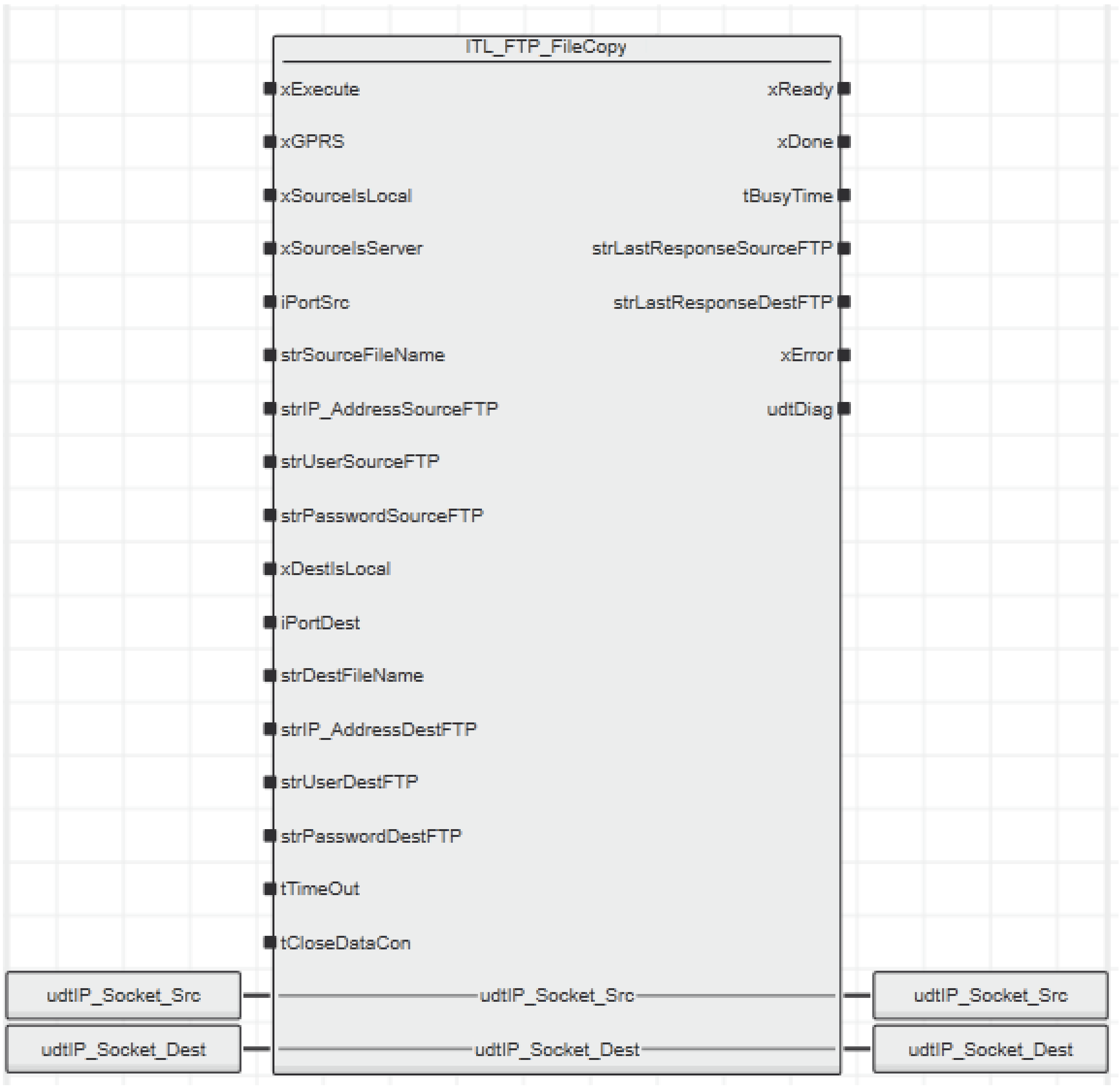
5 ITL_FTP

FTP is a communication protocol for transmitting files between two different computer systems. Transmission takes place according to the client and server principle. An FTP server makes files available to an FTP client. The FTP client can upload, download and delete files on the FTP server. It is possible to manage files conveniently with an appropriate FTP client.

5.1 ITL_FTP_FileCopy

This function block is used to copy a file from a source FTP server to a destination FTP server.

5.1.1 Function block call



5.1.2 Input parameters

Name	Type	Description
xExecute	BOOL	Rising edge: Executes the function block.
xGPRS	BOOL	Sets the internal timeouts to a value that is appropriate for GPRS or for other slow communication with slow FTP servers (IP_Socket timeouts 100s and general timeout 900s). For using custom timeouts, leave xGPRS on FALSE and set the tTimeout input. Also you can set the IP_Socket timeouts directly in the IP_Socket structure before starting the function block (udtIP_Socket_Src.tConnectTimeout, udtIP_Socket_Src.tReceiveTimeout, udtIP_Socket_Src.tSendTimeout and the same for udtIP_Socket_Dest). For more details refer to the IP_COM documentation.
xSourceIsLocal	BOOL	TRUE if the source FTP is the FTP of the PLC himself. Is necessary only on ECLR PLCs.
xSourceIsServer	BOOL	TRUE if source is passive (server) for FTP data connection, FALSE if destination is passive (server).
iPortSrc	INT	Port of source FTP server.
strSourceFileName	STRING	File name of source file.
strIP_AddressSourceFTP	STRING	IP address of source FTP server.
strUserSourceFTP	STRING	Username for source FTP server.
strPasswordSourceFTP	STRING	Password of the selected username for source FTP server.
xDestIsLocal	BOOL	TRUE if the remote FTP is the FTP of the PLC himself. Is necessary only on ECLR PLCs.
iPortDest	INT	Port of destination FTP server.
strDestFileName	STRING	File name of destination file.
strIP_AddressDestFTP	STRING	IP address of destination FTP server.
strUserDestFTP	STRING	Username for destination FTP server.
strPasswordDestFTP	STRING	Password of the selected username for destination FTP server.
tTimeout	TIME	Time out time for complete process.
tCloseDataCon	TIME	Wait time at the end of the procedure, before closing the data connection.

5.1.3 Output parameters

Name	Type	Description
xReady	BOOL	FALSE: The function block is executing services. TRUE: The function block is ready to execute services.
xDone	BOOL	TRUE: Request was sent and the response from communication partner received successfully.
tBusyTime	TIME	Used time for operation.
strLastResponseSourceFTP	STRING	Last response from source FTP server.
strLastResponseDestFTP	STRING	Last response from destination FTP server.
xError	BOOL	TRUE: An error has occurred. For details refer to wDiagCode and wAddDiagCode.
udtDiag	ITL_FTP_UDT_FILE_COPY_DIAG	Diagnostic structure that contains all diagnostic information.

5.1.4 Inout parameters

Name	Type	Description
udtIP_Socket_Src	IPC_UDT_IP_SOCKET	Communication structure to create a network communication with IPC_Socket from IP_Com.
udtIP_Socket_Dest	IPC_UDT_IP_SOCKET	Communication structure to create a network communication with IPC_Socket from IP_Com.

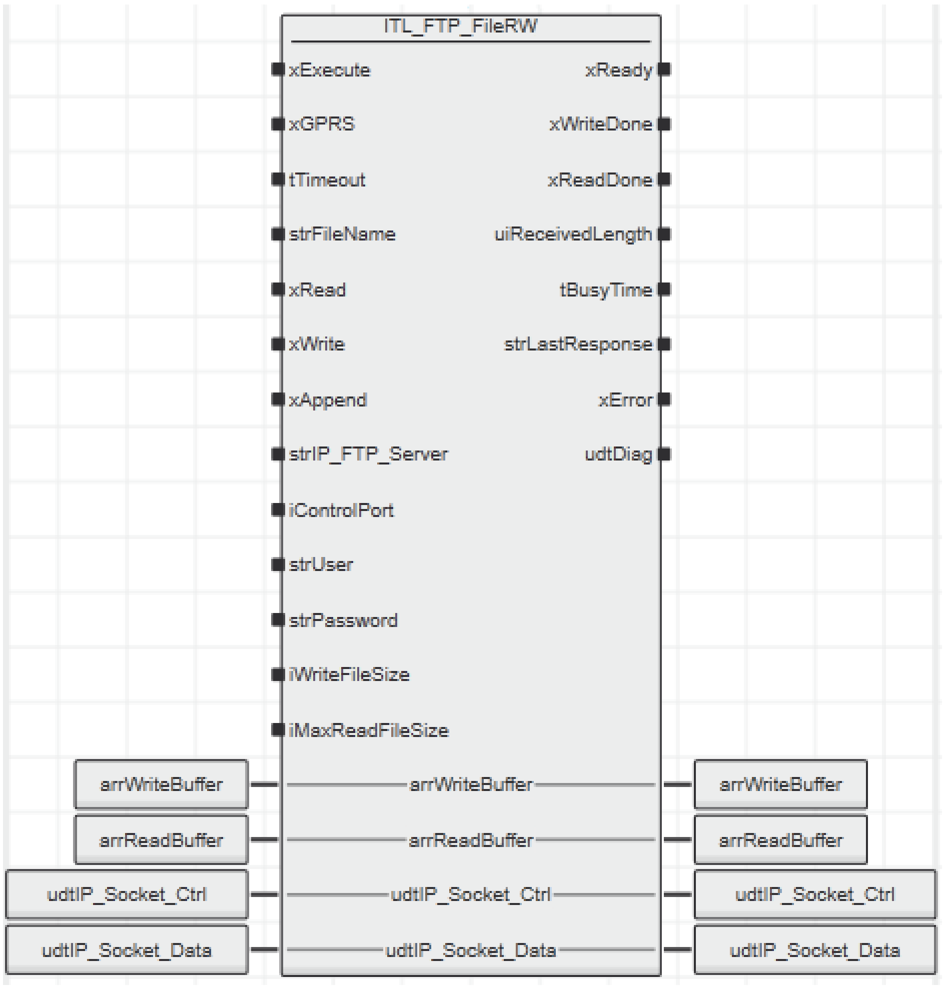
5.1.5 Diagnosis

wDiagCode	wAddDiagCode	Description
16#0000	16#0000	Function block is deactivated.
16#8100	16#0000	Function block is in the initialization phase.
16#8200	16#0000	Wait until socket is not occupied anymore.
16#8300	16#0000	Function block is running.
16#C101	IPC_SOCKET error code.	Source FTP : error during IP connection.
16#C111	IPC_SOCKET error code.	Destination FTP : error during IP connection.
16#C302	IPC_SOCKET error code.	Source FTP : data (user) could not be send.
16#C312	IPC_SOCKET error code.	Destination FTP : data (user) could not be send.
16#C303	IPC_SOCKET error code.	Source FTP : data (password) could not be send.
16#C313	IPC_SOCKET error code.	Destination FTP : data (password) could not be send.
16#C304	IPC_SOCKET error code.	Source FTP : data (Type I) could not be send.
16#C314	IPC_SOCKET error code.	Destination FTP : data (Type I) could not be send.
16#C305	IPC_SOCKET error code.	Source FTP : data (PORT) could not be send.
16#C315	IPC_SOCKET error code.	Destination FTP : data (PASV) could not be send.
16#C306	IPC_SOCKET error code.	Source FTP : data (RETR) could not be send.
16#C316	IPC_SOCKET error code.	Destination FTP : data (STOR) could not be send.
16#C307	IPC_SOCKET error code.	Source FTP : data (QUIT) could not be send.
16#C317	IPC_SOCKET error code.	Destination FTP : data (QUIT) could not be send.
16#C30F	16#0000	Source FTP : error from FTP server, see strLastResponseSourceFTP.
16#C31F	16#0000	Destination FTP : error from FTP server, see strLastResponseDestFTP.
16#C3FE	16#0000	Timeout. Execution took longer than the connected tTimeout.

5.2 ITL_FTP_FileRW

This function block is used to read, write or append a file on a FTP server.

5.2.1 Function block call



5.2.2 Input parameters

Name	Type	Description
xExecute	BOOL	Rising edge: Executes the function block.
xGPRS	BOOL	Sets the internal timeouts to a value that is appropriate for GPRS or for other slow communication or for slow FTP servers.
tTimeout	TIME	Time out time for complete process.
strFileName	STRING	Name of the file.
xRead	BOOL	Reads out the selected file.
xWrite	BOOL	Writes the selected file.
xAppend	BOOL	Appends to the selected file.
strIP_FTP_Server	STRING	IP address of FTP server.
iControlPort	INT	Port of the FTP server.
strUser	STRING	Username for the FTP server.
strPassword	STRING	Password of the selected username for the FTP server.
iWriteFileSize	INT	Number of characters to be written in bytes. A maximum of only 32767 bytes can be written! Please observe that the value of the iWriteFileSize parameters may not exceed the size of array of byte (bFirstByteOfWriteBuffer). Example: bFirstByteOfWriteBuffer[1..500] iWriteFileSize may not be larger than 500.
iMaxReadFileSize	INT	Maximum size of the receive buffer. Please observe that the value of the iReadFileSize parameter may not exceed the size of the array of byte bFirstByteOfReadBuffer), Example: bFirstByteOfReadBuffer[1..800] iMaxReadFileSize may not be larger than 800.

5.2.3 Output parameters

Name	Type	Description
xReady	BOOL	FALSE: The function block is executing services. TRUE: The function block is ready to execute services.
xWriteDone	BOOL	Is 1 cycle TRUE, when xWrite/ xAppend was executed successfully.
xReadDone	BOOL	Is 1 cycle TRUE, when xRead was executed successfully.
uiReceivedLength	UINT	Used time for operation.
tBusyTime	TIME	Used time for operation.
strLastResponse	STRING	Last response from the FTP server.
xError	BOOL	TRUE: An error has occurred. For details refer to wDiagCode and wAddDiagCode.
udtDiag	ITL_FTP_UDT_FILE_FRW	Diagnostic structure that contains all diagnostic information.

5.2.4 Inout parameters

Name	Type	Description
arrWriteBuffer	ITL_FTP_ARR_BUFFER	Array of BYTE for writing into a file.
arrReadBuffer	ITL_FTP_ARR_BUFFER	Array of BYTE for reading from a file.
udtIP_Socket_Ctrl	IPC_UDT_IP_SOCKET	Communication structure to create a network communication with IPC_Socket from IP_Com.
udtIP_Socket_Data	IPC_UDT_IP_SOCKET	Communication structure to create a network communication with IPC_Socket from IP_Com.

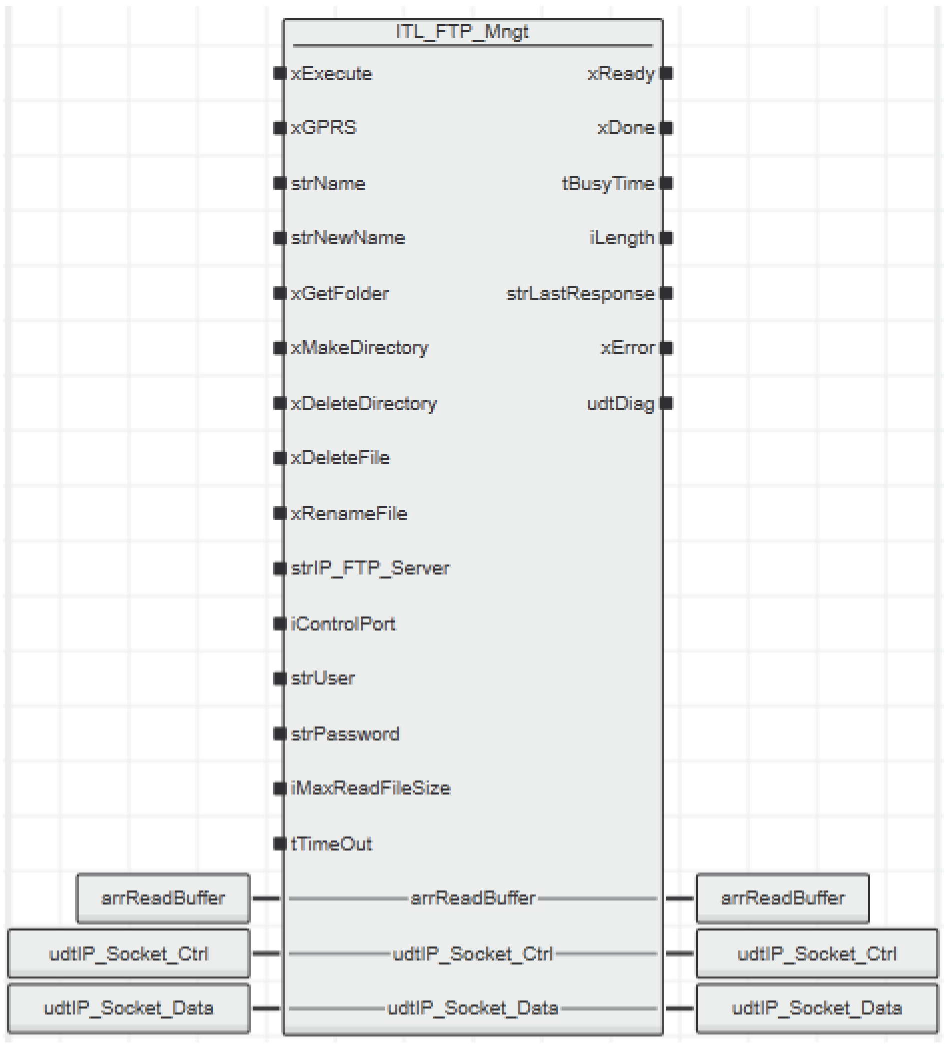
5.2.5 Diagnosis

wDiagCode	wAddDiagCode	Description
16#0000	16#0000	Function block is deactivated.
16#8100	16#0000	Function block is in the initialization phase.
16#8200	16#0000	Wait until socket is not occupied anymore.
16#8300	16#0000	Function block is running.
16#C100	16#0000	Only one signal of xRead, xWrite and xAppend at the same time allowed.
16#C102		Permitted length of variabled excedeed.
	16#0000	strUser is too long.
	16#0001	strFileName is too long.
16#C301	Diagnostic code of IP_SEND function block.	Data (user) could not be send.
16#C303	Diagnostic code of IP_SEND function block.	Data (password) could not be send.
16#C304	Diagnostic code of IP_SEND function block.	Data (Type I) could not be send.
16#C305	Diagnostic code of IP_SEND function block.	Data (PORT) could not be send.
16#C306	Diagnostic code of IP_SEND function block.	Data (RETR) could not be send.
16#C307	Diagnostic code of IP_SEND function block.	Data (STOR/ APPE) could not be send.
16#C308	Diagnostic code of IP_SEND function block.	Data (QUIT) could not be send.
16#C310	Diagnostic code of IP_CONNECT function block.	Connection could not be established.
16#C311	16#0000	Append is not supported by the FTP server.
16#C30F	Receive code from FTP server.	FTP server reports an error, see strLastResponse.
16#C3FE	Internal step.	Timeout. Execution took longer than the connected tTimeout.

5.3 ITL_FTP_Mngt

This function block is used to manage files and directories on the FTP server. For files it is possible to rename and delete them. For directories is it possible to create and delete them. It is also possible to get additional information about the directory.

5.3.1 Function block call



5.3.2 Input parameters

Name	Type	Description
xExecute	BOOL	Rising edge: Executes the function block.
xGPRS	BOOL	Sets the internal timeouts to a value that is appropriate for GPRS or for other slow communication with slow FTP servers (IP_Socket timeouts 100s and general timeout 900s). For using custom timeouts, leave xGPRS on FALSE and set the tTimeout input. Also you can set the IP_Socket timeouts directly in the IP_Socket structure before starting the function block (udtIP_Socket_Src.tConnectTimeout, udtIP_Socket_Src.tReceiveTimeout, udtIP_Socket_Src.tSendTimeout and the same for udtIP_Socket_Dest). For more details refer to the IP_COM documentation.
strName	STRING	Name of the file.
strNewName	STRING	New name of the new file.
xGetFolder	BOOL	Get information about the selected folder.
xMakeDirectory	BOOL	Create a new folder with the selected name.
xDeleteDirectory	BOOL	Delete the selected folder.
xDeleteFile	BOOL	Delete the selected file.
xRenameFile	BOOL	Rename the selected file with the name strNewName.
strIP_FTP_Server	STRING	IP address of FTP server.
iControlPort	INT	Port of the FTP server.
strUser	STRING	Username for the FTP server.
strPassword	STRING	Password of the selected username for the FTP server.
iMaxReadFileSize	INT	Maximum size of the receive buffer. Please observe that the value of the iReadFileSize parameter may not exceed the size of the array of byte bFirstByteOfReadBuffer), Example: bFirstByteOfReadBuffer[1..800] iMaxReadFileSize may not be larger than 800.
tTimeout	TIME	Time out time for complete process.

5.3.3 Output parameters

Name	Type	Description
xReady	BOOL	FALSE: The function block is executing services. TRUE: The function block is ready to execute services.
xDone	BOOL	TRUE: Request was sent and the response from communication partner received successfully.
tBusyTime	TIME	Used time for operation.
iLength	INT	Number of received bytes.
strLastResponse	STRING	Last response from the FTP server.
xError	BOOL	TRUE: An error has occurred. For details refer to wDiagCode and wAddDiagCode.
udtDiag	ITL_FTP_UDT_FILE_FRW	Diagnostic structure that contains all diagnostic information.

5.3.4 Inout parameters

Name	Type	Description
arrReadBuffer	ITL_FTP_ARR_BUFFER	Array of BYTE for reading from a file.
udtIP_Socket_Ctrl	IPC_UDT_IP_SOCKET	Communication structure to create a network communication with IPC_Socket from IP_Com.
udtIP_Socket_Data	IPC_UDT_IP_SOCKET	Communication structure to create a network communication with IPC_Socket from IP_Com.

5.3.5 Diagnosis

wDiagCode	wAddDiagCode	Description
16#0000	16#0000	Function block is deactivated.
16#8100	16#0000	Function block is in the initialization phase.
16#8200	16#0000	Wait until socket is not occupied anymore.
16#8300	16#0000	Function block is running.
16#C100	16#0000	Only one signal of xGetFolder, xMakeDirectory, xDeleteDirectory, xDeleteFile and xRenameFile at the same time allowed.
16#C301	Diagnostic code of IP_SEND function block.	Data (user) could not be send.
16#C303	Diagnostic code of IP_SEND function block.	Data (password) could not be send.
16#C304	Diagnostic code of IP_SEND function block.	Data (Type I) could not be send.
16#C305	Diagnostic code of IP_SEND function block.	Data (PORT) could not be send.
16#C306	Diagnostic code of IP_SEND function block.	Data (RETR) could not be send.
16#C310	Diagnostic code of IP_CONNECT function block.	Connection could not be established.
16#C30F	Receive code from FTP server.	FTP server reports an error, see strLastResponse.
16#C3FE	Internal step.	Timeout. Execution took longer than the connected tTimeout.

5.4 Startup example

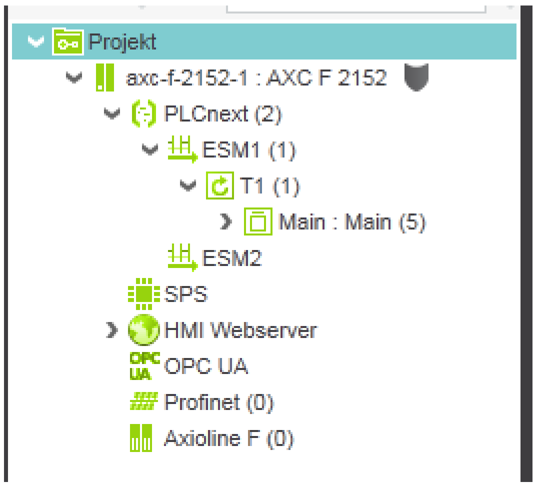
For the startup instruction please find the following example:

- ITL_8_EXA_ITL_FTP.pcwex

Plant

For this example, the following hardware is used:

- AXC F 2152 (2404267)

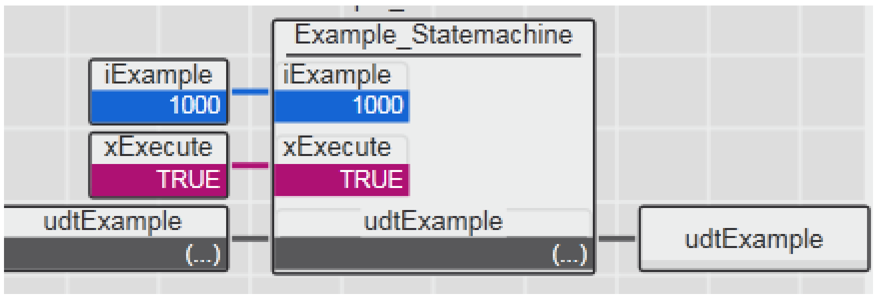


Projekt
axc-f-2152-1 : AXC F 2152
PLCnext (2)
ESM1 (1)
T1 (1)
Main : Main (5)
ESM2
SPS
HMI Webserver
OPC UA
Profinet (0)
Axioline F (0)

The project shows one startup example for each function of the function blocks. They can be found inside the Example_Statemachine function block. There are state machines for every step we have to take care of when using one functionality.

For starting the example, we have to go in every code sheet and adjust the setup of the function blocks (server and access data) for our settings.

Then we can execute the desired example and start the function block by setting xExecute to TRUE.



The following function can be executed:

Function	iExample	Codesheet
FileRW reading a file.	1000	FileRW_Read
FileRW writing a file.	2000	FileRW_Write
FileRW append to a file.	3000	FileRW_Append
FileCopy copy a file from a source FTP server to a destination FTP server.	4000	FileCopy
Mngt get additional information about a folder.	5000	Mngt_GetFolder

Mngt create a directory.	6000	Mngt_MakeDirectory
Mngt delete a directory.	7000	Mngt_DeleteDirectory
Mngt delete a file.	8000	Mngt_DeleteFile
Mngt rename a file to another name.	9000	Mngt_RenameFile

The structure of every example is the same. First we have to initialize the used function block and set every input variable.

```
0 : (*init*)
    (* This is a setup for a local FileZilla FTP server. strFileName, strIP_FTP_se
       strUser and strPassword have to be adjusted to the used FTP server.*)
    udtExample.udtFileRW.xExecute := FALSE;
    udtExample.udtFileRW.xGPRS := FALSE;
    udtExample.udtFileRW.tTimeOut := T#50s;
    udtExample.udtFileRW.strFileName := 'TestFileSource.txt';
    udtExample.udtFileRW.xRead := TRUE;
    udtExample.udtFileRW.xWrite := FALSE;
    udtExample.udtFileRW.xAppend := FALSE;
    udtExample.udtFileRW.strIP_FTP_Server := '192.168.178.55';
    udtExample.udtFileRW.iControlPort := 21;
    udtExample.udtFileRW.strUser := 'Daniel';
    udtExample.udtFileRW.strPassword := '123456';
    udtExample.udtFileRW.iWriteFileSize := 0;
    udtExample.udtFileRW.iMaxReadFileSize := 80;

    udtExample.iState := 10;
```

Then we have to wait for our function block to be ready. The function block ignores the xExecute when it is not ready. As far as the function block is ready, we set xExecute.

```
10 :
    (*wait for function block to be ready*)
    IF udtExample.udtFileRW.xReady THEN
        udtExample.udtFileRW.xExecute := TRUE;
        udtExample.iState := 20;
    END_IF;
```

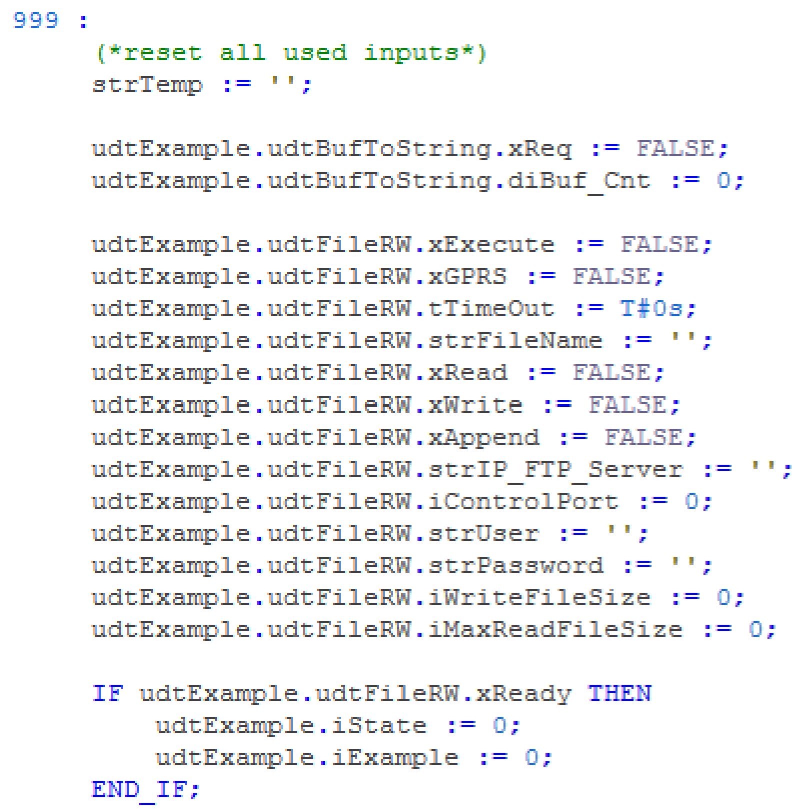
If execution is done, the xDone (in this case xReadDone) output is TRUE for one cycle. Now we can process the data we just read out. For this we use a BUF_TO_STRING function block and execute it.

```
20 :
    (*wait for reading finish*)
    IF udtExample.udtFileRW.xReadDone THEN
        (* convert the received buffer (arrReadBuffer) to string *)
        udtExample.udtBufToString.diBuf_Cnt := TO_DINT(udtExample.udtFileRW.uiRecei
        udtExample.udtBufToString.xReq := TRUE;
        udtExample.iState := 30;
    END_IF;
```

After this function is complete, we copy the converted string and can use it for any purpose.

```
30 :
    (*wait for BUF_TO_STRING to finish*)
    IF udtExample.udtBufToString.xDone THEN
        (*now we can copy the string-value to our local variable and safe the read
        strTemp := udtExample.udtBufToString.strDestination;
        udtExample.iState := 999;
    END_IF;
```

Finally, in order to execute the function block again, all inputs must be reset.



```
999 :
     (*reset all used inputs*)
     strTemp := '';

     udtExample.udtBufToString.xReq := FALSE;
     udtExample.udtBufToString.diBuf_Cnt := 0;

     udtExample.udtFileRW.xExecute := FALSE;
     udtExample.udtFileRW.xGPRS := FALSE;
     udtExample.udtFileRW.tTimeOut := T#0s;
     udtExample.udtFileRW.strFileName := '';
     udtExample.udtFileRW.xRead := FALSE;
     udtExample.udtFileRW.xWrite := FALSE;
     udtExample.udtFileRW.xAppend := FALSE;
     udtExample.udtFileRW.strIP_FTP_Server := '';
     udtExample.udtFileRW.iControlPort := 0;
     udtExample.udtFileRW.strUser := '';
     udtExample.udtFileRW.strPassword := '';
     udtExample.udtFileRW.iWriteFileSize := 0;
     udtExample.udtFileRW.iMaxReadFileSize := 0;

     IF udtExample.udtFileRW.xReady THEN
         udtExample.iState := 0;
         udtExample.iExample := 0;
     END_IF;
```

5.5 Data types

```
TYPE
    (*Array for the last received FTP requests*)
    ITL_FTP_ARR_REQUESTS : ARRAY [1..10] OF STRING;

    (*Diagnostic struct for ITL_FTP_FileCopy*)
    ITL_FTP_UDT_FILE_COPY_DIAG : STRUCT
        wDiagCode : WORD; (*Diagnostic code*)
        wAddDiagCode : WORD; (*Additional diagnostic code*)
        iState_Src : INT; (*State of soruce statemachine*)
        iState_Dest : INT; (*State of destination statemachine*)
    END_STRUCT;

    (*Diagnostic struct for ITL_FTP_FileRW and ITL_FTP_Mngt*)
    ITL_FTP_UDT_FILE_RW_DIAG : STRUCT
        wDiagCode : WORD; (*Diagnostic code*)
        wAddDiagCode : WORD; (*Additional diagnostic code*)
        iState : INT; (*State of statemachine*)
    END_STRUCT;
    (*Buffer for reading and writing*)
    ITL_FTP_ARR_BUFFER : ARRAY[1..16000] OF BYTE;
END_TYPE
```

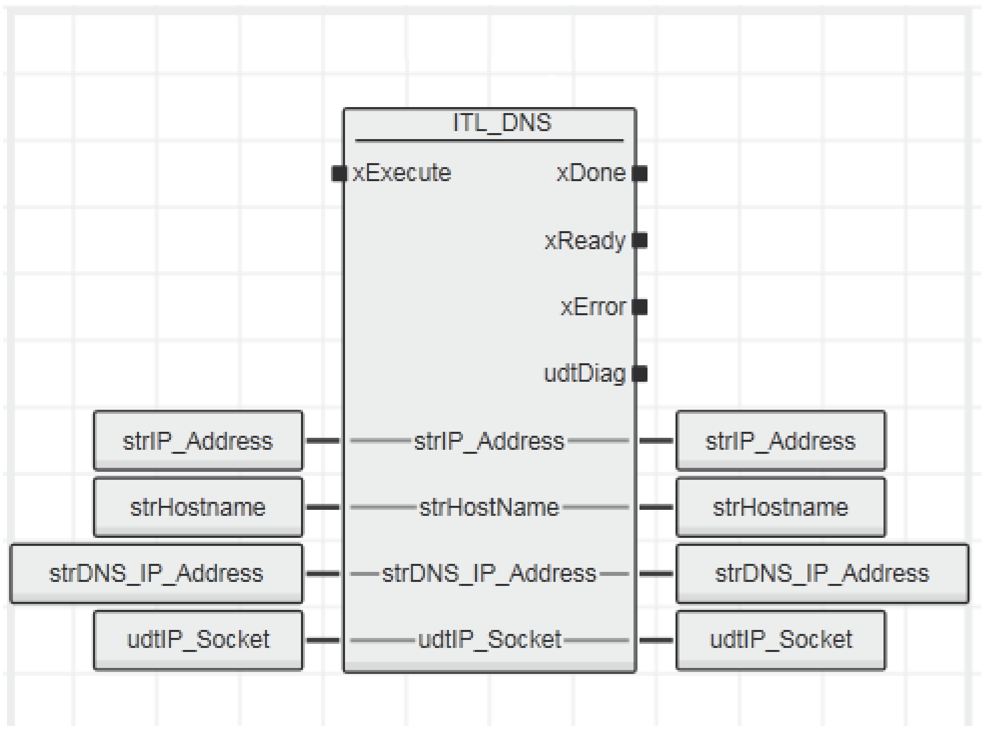
6 ITL_DNS

This function block enables a host name to be read out to an IP address using a DNS server

The Domain Name System (DNS) is one of the most important services in the internet. Its main task is answering queries on name resolution. Similar to directory enquiries, DNS should state

- the host name (the address in the internet, e.g. de.wikipedia.org) when queried
- and return the corresponding IP address (the connection number, e.g. 91.198.174.2)

6.1 Function block call



6.2 Input parameters

Name	Type	Description
xExecute	BOOL	Rising edge: Executes the function block.

6.3 Output parameters

Name	Type	Description
xDone	BOOL	TRUE: Request was sent and the response from communication partner received successfully.
xReady	BOOL	FALSE: The function block is executing services. TRUE: The function block is ready to execute services.
xError	BOOL	TRUE: An error has occurred. For details refer to wDiagCode and wAddDiagCode.
udtDiag	ITL_DNS_DIAG	Diagnostic structure.

6.4 Inout parameters

Name	Type	Description
strIP_Address	STRING	After successful resolution of the host name, the IP address is output here.
strHostName	STRING	The host name to be resolved is specified here, e.g. www.phoenixcontact.com
strDNS_IP_Address	STRING	The IP address of the DNS server is transferred here. For example: STRING# '192.168.0.1'
udtIP_Socket	IPC_UDT_IP_SOCKET	Communication structure to create a network communication with IPC_Socket from IP_Com.

6.5 Diagnosis

wDiagCode	wAddDiagCode	Description
16#0000	16#0000	Function block is deactivated.
16#8000	16#0000	Function block is in regular operation.
16#8200	16#0000	Wait until socket is not occupied anymore.
16#C110		Invalid Input.
	16#0001	strHostName.
	16#0002	strDNS_IP_Address.
16#C301		Error during communication with the DNS server.
	16#0001	DNS request us too large.
	16#0002	Host name could not be found.
	16#0003	Invalid value.
	16#0004	Read only.
	16#0005	General error during resolution of the host name.
	16#0006	No answer from DNS server.
	16#0007	Connection to server could not be established.
	16#0008	Host address could not be found in answer.
16#C330		Error at UDP_SOCKET function block.
	16#xxxx	Refer to appendix.
16#C331		Error at UDP_RECEIVE function block.
	16#xxxx	Refer to appendix.
16#C332		Error at UDP_SEND function block.
	16#xxxx	Refer to appendix.
16#C410	16#0000	Timeout while initializing.
16#C414	16#0000	Timeout while sending (UDP_SEND).
16#C415	16#0000	Timeout while receiving (UDP_RECEIVE).

6.6 Startup example

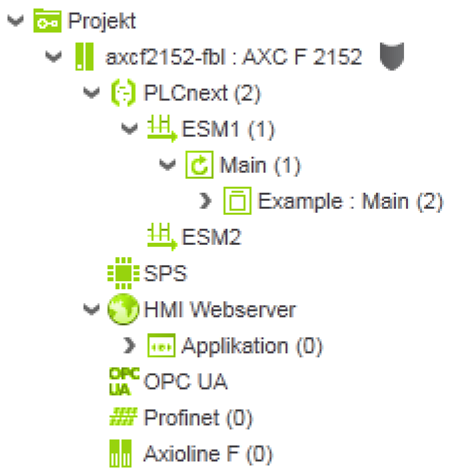
For the startup instruction please find the following example:

- ITL_8_EXA_ITL_DNS.pcwex

Plant

For this example, the following hardware is used:

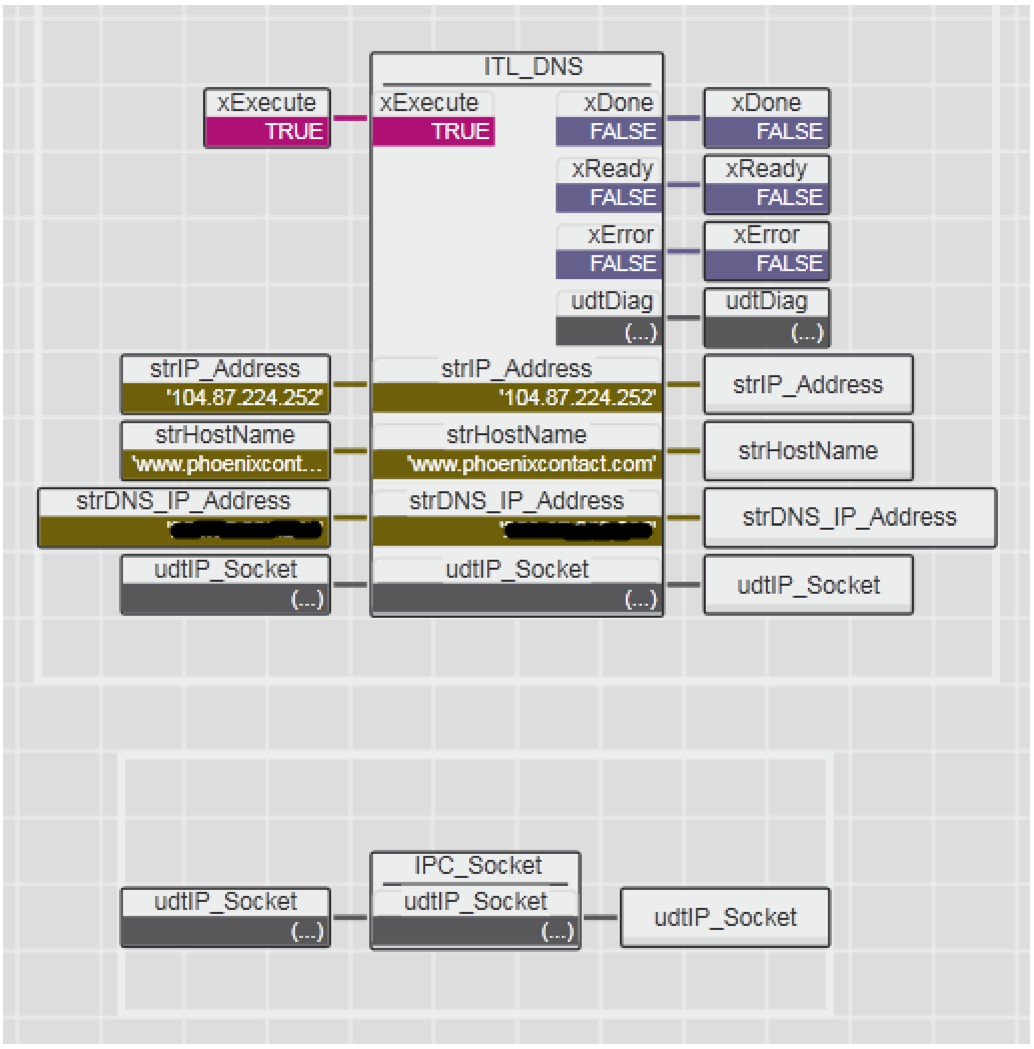
- AXC F 2152 (2404267)



This project shows one example for the startup of ITL_DNS function block.

Enter the strHostname, the strDNS_IP_Address and set xReq to TRUE.
If the xDone output is TRUE, the process is completed successfully and the IP-Address is shown in the InOut strIP_Address.

Note: Please replace data with your own valid data!



6.7 Data types

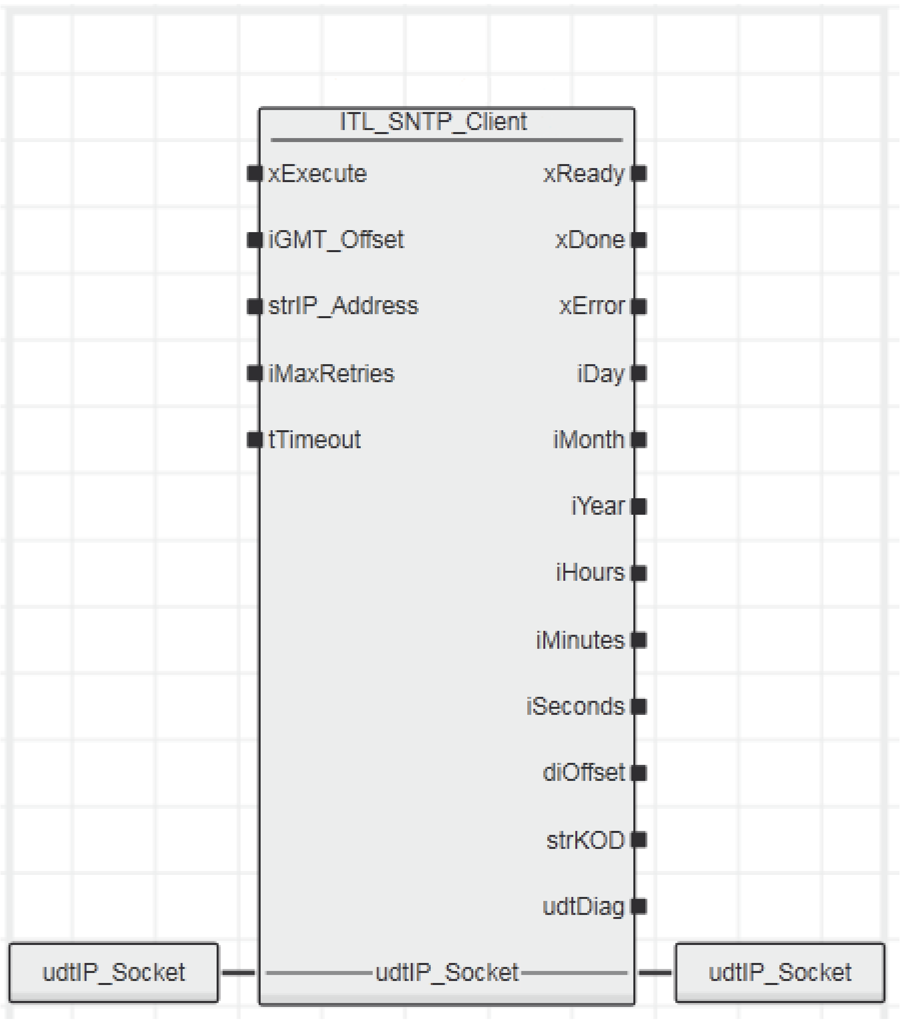
```
TYPE
    (*Array for the response of the DNS server*)
    ITL_DNS_ARR_B_1_500    :    ARRAY [1..500] Of BYTE;
    (*Array for the buffer of the host address*)
    ITL_DNS_ARR_B_1_80     :    ARRAY [1..80] OF BYTE;
    ITL_DNS_UDT_QUERRY  :   STRUCT (*Struct for the DNS query*)
        wTransactionID          :       WORD;
        wFlags                  :       WORD;
        (*
        0... .... .... .... = Response: Message is a query
        .000 0... .... .... = Opcode: Standard query (0)
        .... ..0. .... .... = Truncated: Message is not truncated
        .... ...1 .... .... = Recursion desired: Do query recursively
        .... .... .0.. .... = Z: reserved (0)
        .... .... ...0 .... = Non-authenticated data OK:
                            Non-authenticated data is unacceptable*)
        wQuestions              :       WORD;
        wAnswer_RRs             :       WORD;
        wAuthority_RRs          :       WORD;
        wAdditional_RRs         :       WORD;
    END_STRUCT;

    ITL_DNS_UDT_SOCKET_RCV  :   STRUCT (*Struct for received data*)
        iCnt                    :       INT;
        udtDNS_Querry           :       ITL_DNS_UDT_QUERRY;
        arrResponse             :       ITL_DNS_ARR_B_1_500;
    END_STRUCT;

    ITL_DNS_UDT_DIAG    :   STRUCT  (*Diag struct*)
        wDiagCode       :   WORD;
        wAddDiagCode    :   WORD;
        iState          :   INT;
        iStateRcv       :   INT;
    END_STRUCT;
END_TYPE
```

7 ITL_SNTP_Client

The SNTP_Client block determines the current time of an (S)NTP server via the SNTP protocol and makes this available at its outputs for further processing.

Please note, the function block can not be used for GPRS communication!

7.1 Function block call



7.2 Input parameters

Name	Type	Description
xExecute	BOOL	Rising edge: Executes the function block.
iGMT_Offset	INT	GMT offset for local adaptation.
strIP_Address	STRING	IP address of the SNTP server (e.g., “192.168.0.1”).
iMaxRetries	INT	Number of permitted failed attempts.
tTimeout	TIME	Timeout between the individual attempts.

7.3 Output parameters

Name	Type	Description
xReady	BOOL	FALSE: The function block is executing services. TRUE: The function block is ready to execute services.
xDone	BOOL	TRUE: Request was sent and the response from communication partner received successfully.
xError	BOOL	TRUE: An error has occurred. For details refer to wDiagCode and wAddDiagCode.
iDay	INT	Date of the server, day.
iMonth	INT	Date of the server, month.
iYear	INT	Date of the server, year.
iHours	INT	Date of the server, hours.
iMinutes	INT	Date of the server, minutes.
iSeconds	INT	Date of the server, seconds.
diOffset	DINT	Date of the server, seconds since 1.1.1970.
strKOD	STRING	Kiss of death codes.
udtDiag	ITL_SNTP_UDT_DIAG	Diagnosis structure.

7.4 Inout parameters

Name	Type	Description
udtIP_Socket	IPC_UDT_IP_SOCKET	Communication structure to create a network communication with IPC_Socket from IP_Com.

7.5 Diagnosis

wDiagCode	wAddDiagCode	Description
16#0000	16#0000	Function block is deactivated.
16#8000	16#0000	Function block is in regular operation.
16#8200	16#0000	Wait until socket is not occupied anymore.
16#C110		Invalid Input.
	16#0001	strDIP_Address.
16#C001		Error during communication with the DNS server.
	16#0001	Length of response telegram is not equal to 48 bytes.
	16#0002	Server flag is not set mode <> 4.
	16#0003	Server time is smaller than 1.1.1970.
	16#0004	RTimeOut - server did not respond within specified time..
16#C330		Error at UDP_SOCKET function block.
	16#xxxx	Refer to appendix.
16#C331		Error at UDP_RECEIVE function block.
	16#xxxx	Refer to appendix.
16#C332		Error at UDP_SEND function block.
	16#xxxx	Refer to appendix.
16#C410	16#0000	Timeout while initializing.
16#C414	16#0000	Timeout while sending (UDP_SEND).
16#C415	16#0000	Timeout while receiving (UDP_RECEIVE).

7.6 Startup example

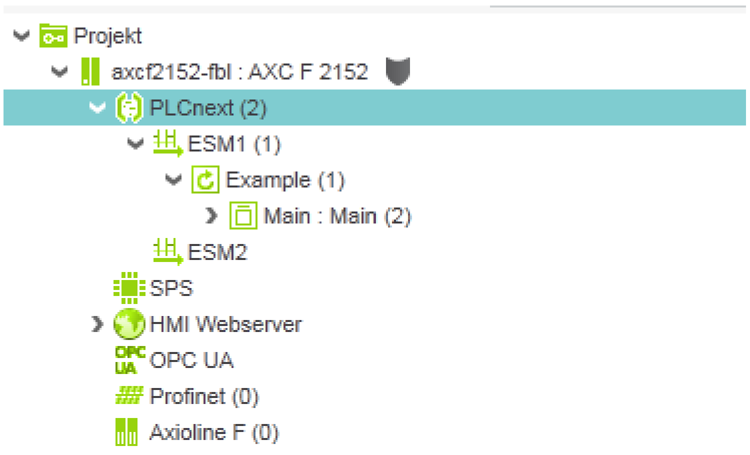
For the startup instruction please find the following example:

- ITL_8_EXA_ITL_SNTP_Client.pcwex

Plant

For this example, the following hardware is used:

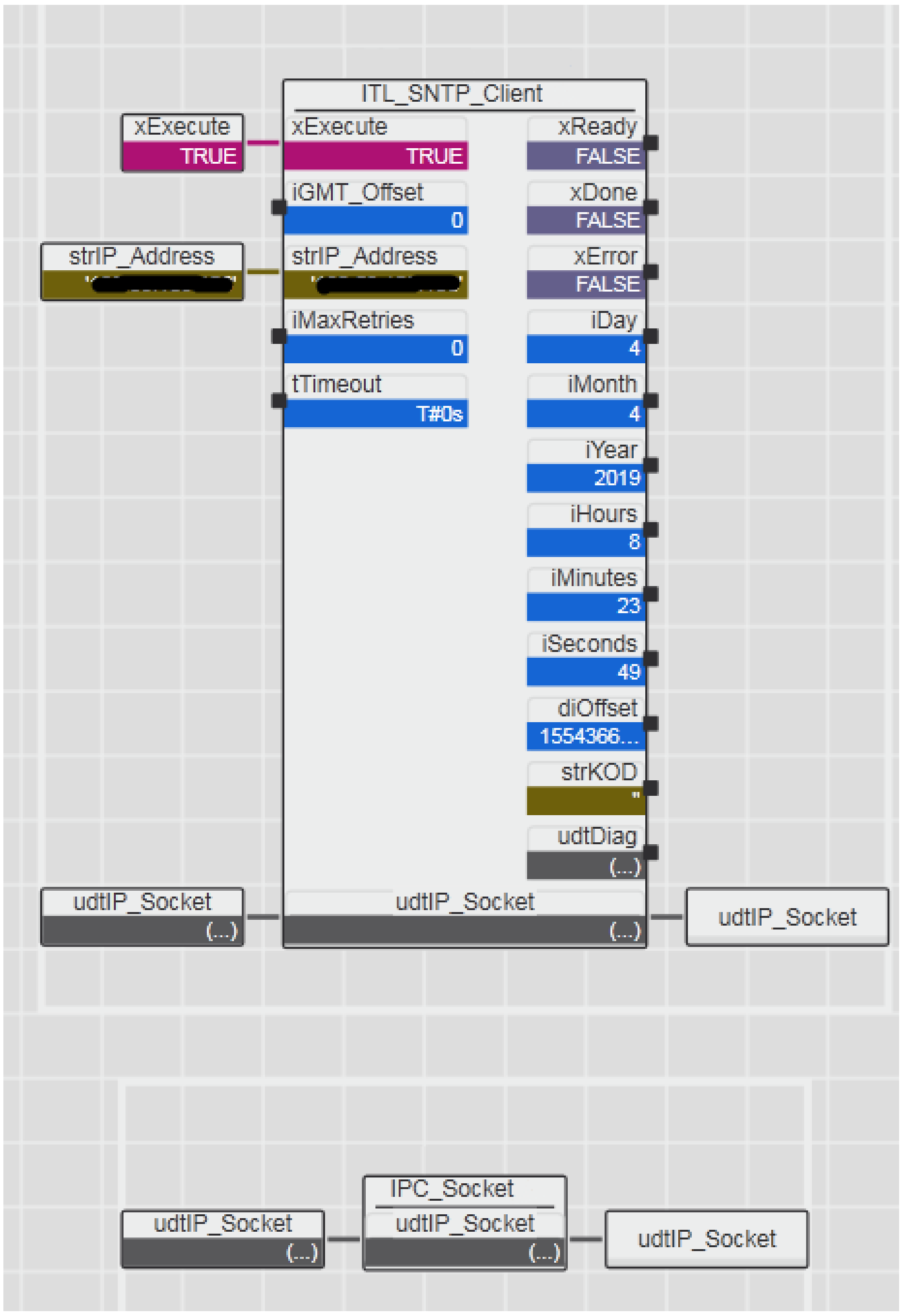
- AXC F 2152 (2404267)



This project shows one example for the startup of ITL_SNTP_Client function block.

Enter the strIP_Address and set xExecute to TRUE.
If the output xDone is TRUE, the process is successfully completed and the current time is shown in the outputs.

Note: Please replace data with yout own valid data!



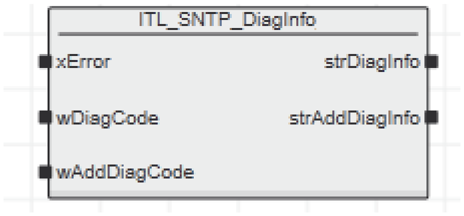
7.7 Data types

```
TYPE
    ITL_SNTP_UDT_DIAG    : STRUCT (*Diag struct for ITL_SNTP_Client*)
        wDiagCode        : WORD;
        wAddDiagCode     : WORD;
        iState           : INT; (*current state*)
        iRetries         : INT; (*current made retries*)
    END_STRUCT;
END_TYPE
```

8 ITL_SNTP_DiagInfo

In case of an error at the ITL_SNTP_Client, this block shows the diagnostics of the block as a text. This code is neither write nor read protected, so the code can be adapted to the individual demands. Diagnostic text can be added, edited or translated. The text output of strDiagInfo and strAddDiagInfo is limited to 80 characters.

8.1 Function block call



8.2 Input parameters

Name	Type	Description
xError	BOOL	xError output of ITL_SNTP_Client function block.
wDiagCode	WORD	wDiagCode from ITL_SNTP_Client function block (udtDiag.wDiagCode).
wAddDiagCode	WORD	wAddDiagCode from ITL_SNTP_Client function block (udtDiag.wAddDiagCode).

8.3 Output parameters

Name	Type	Description
strDiagInfo	STRING	Diagnosis information as text.
strAddDiagInfo	STRING	Extended diagnosis information as text.

9 ITL_SMTP_Client

E-mails can be sent with this function block. SMTP is used as transmission protocol. A file can also be attached to an e-mail that has been stored on the CPU's flash card. The SMTP standard port 25 is used as TCP-IP port.

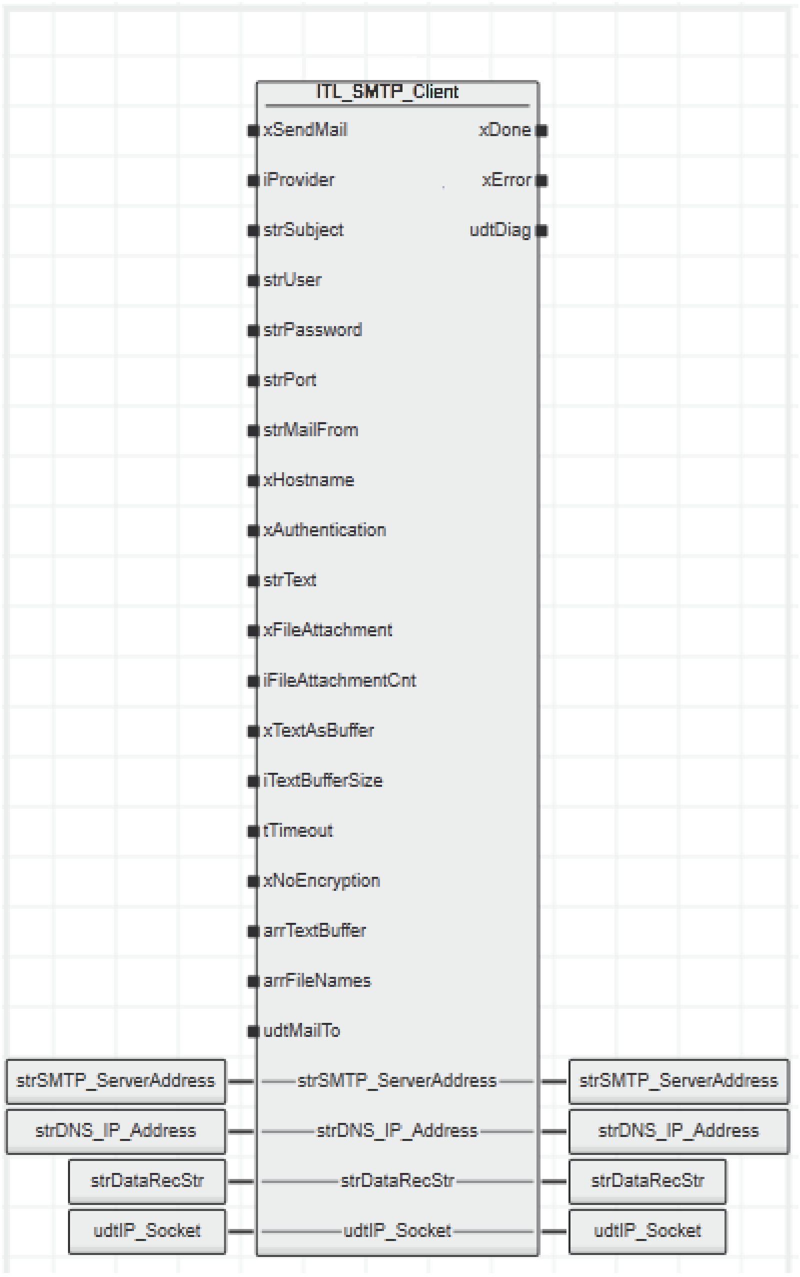
Please note, the function block can not be used for GPRS communication!

9.1 Releases

This function block version is approved for the following provider:

Port	G-Mail	GMX	Web.de	Yahoo	hotmail	Freenet	NoEncyryption
25	-	-	-	-	-	-	+
465	+	+	-	+	-	+	-
587	+	+	+	+	+	+	+
iProvider	1	1	1	1	1	0	1

9.2 Function block call



9.3 Input parameters

Name	Type	Description
xSendMail	BOOL	The email is sent in the case of a positive edge. The TRUE signal may only be set to FALSE when **xDone** or **xError** has been received.
iProvider	INT	Select provider: • 0: Freenet.net • 1: Other provider Refer to table Releases
strSubject	STRING	Subject line of the email, maximum of 80 characters.
strUser	STRING	User's identification if necessary.
strPassword	STRING	E-mail account password
strPort	STRING	Port number of the Server. Init value = 25.
strMailFrom	STRING	The sender's mail address. The sender's mail account address must be stated here. In most cases this is a security function of the mail server.
xHostname	BOOL	The block can be resolved from a host name to an IP address over a DNS server (e.g. smtp.web.de). • FALSE = in the case of a **strSMTP_ServerAddress** parameter, only the SMTP server's IP address will be stated (e.g. 10.212.64.3). • TRUE = in the case of a **strSMTP_ServerAddress** parameter, the SMTP server's host name will be stated (e.g. smtp.web.de).
xAuthentication	BOOL	If logon to the SMTP Server is necessary, this parameter must be TRUE.
strText	SMTP_STR_200	If the **xTextAsBuffer** parameter is FALSE, the transferred text is used here as body text. The data type is STRING (200) which is a STRING with a maximum of 200 characters.
xFileAttachment	BOOL	TRUE = file attachment will be sent.
iFileAttachmentCnt	INT	The maximum number of file attachments to be sent is 10. If more files should be sent than given in **arrFileNames**, a new file will be generated.

xTextAsBuffer	BOOL	• FALSE = The **strText** parameter will be used for the text. • TRUE = The **arrTextBuffer** parameter will be used for the text. Note: The number of bytes to be transmitted from the buffer must be transferred to the **iTextBufferSize** parameter. This function on the ILC 1xx with more than 1024 characters is only supported with PC Worx 6.
iTextBufferSize	INT	The number of bytes to be transmitted from the text buffer must be put here.
tTimeOut	TIME	Time monitoring of the Ethernet communication.
xNoEncryption	BOOL	• TRUE: No encryption used • FALSE: Encryption for E-Mail sending **Pay attention of chapter Release!**
arrTextBuffer	ITL_SMTP_ARR_B_1_3200	If the **xTextAsBuffer** parameter is TRUE, the text in the buffer is used as the mail body text. The number of characters that are copied from the buffer is defined by the input **iTextBufferSize**.
arrFileNames	ITL_SMTP_ARR_STR_1_10	Array containing those filenames that are to be transmitted as attachments. A maximum of 10 files may be sent. There is no direct limit on the size of the individual files. If the file does not exist, an empty file is sent
udtMailTo	ITL_SMTP_UDT_EmailAddress	Structure with mail addresses and number of address parameter. For details refer to appendix "Data types".

9.4 Output parameters

Name	Type	Description
xDone	BOOL	TRUE: Request was sent and the response from communication partner received successfully.
xError	BOOL	TRUE: An error has occurred. For details refer to wDiagCode and wAddDiagCode.
udtDiag	ITL_SMTP_UDT_DIAG	Diagnostic structure.

9.5 Inout parameters

Name	Type	Description
strSMTP_ServerAddress	STRING	IP address (e.g. 195.4.92.211) or hostname (e.g. mx.freenet.de) of the SMTP server.
strDNS_IP_Address	STRING	The IP address of the DNS server. Is not necessary when the hostname is parameterized as SMTP_ServerAdress. In most cases the DNS IP = the router IP.
strDataRecStr	ITL_SMTP_STR_200	This parameter is only used for diagnostics purposes. Data received from the SMTP server is entered here.
udtIP_Socket	IPC_UDT_IP_SOCKET	Communication structure to create a network communication with IPC_Socket from IP_Com.

9.6 Diagnosis

wDiagCode	wAddDiagCode	Description
16#0000	16#0000	Function block is deactivated.
16#8000	16#0000	Function block is in regular operation.
16#8200	16#0000	Wait until socket is not occupied anymore.
16#C101		Invalid Input.
	16#0001	strPort.
	16#0002	strSMTP_ServerAddress.
	16#0003	strDNS_IP_Address.
	16#0004	strMailFrom.
	16#0005	strUser.
	16#0006	strPassword.
	16#0007	iProvider.
16#C301		Error during communication with the DNS server.
	16#0001	DNS request us too large.
	16#0002	Host name could not be found.
	16#0003	Invalid value.
	16#0004	Read only.
	16#0005	General error during resolution of the host name.
	16#0006	No answer from DNS server.
	16#0007	Connection to server could not be established.
	16#0008	Host address could not be found in answer.
16#C302		Error during communication with the mail server.
	16#00D3	System status or answer system help
	16#00D6	Help message
	16#00DC	Service is operational
	16#00DD	Service closes transmission channel
	16#00FA	Requested mail action OK, completed
	16#00FB	User not local, will forward to [forwarding path]
	16#0162	Beginning with mail entry; terminate with [CRLF].[CRLF]
	16#01A5	Service not operational, close transmission channel (that can be the reply to any request, if the service knows that it must terminate)
	16#01C2	Requested mail action was not executed: Mailbox not (e.g. mailbox not active)
	16#01C3	Requested action aborted: Local error being processed
	16#01C4	Requested action not executed: insufficient system memory
	16#01F4	Syntax error, instruction not recognized (that also includes a too long command line error)
	16#01F5	Syntax error in parameter or arguments
	16#01F6	Command not implemented
	16#01F7	Unusable command string
	16#01F8	Command parameter not implemented
	16#0226	Requested action not executed: Mailbox not accessible (e.g.: Mailbox was not found, no access)
	16#0227	User not local; please try [forwarding path]

	16#0228	Requested mail action aborted: Assigned memory size was exceeded
	16#0229	Requested action not executed: Mailbox name not allowed (e.g.: Incorrect mailbox syntax)
	16#022A	Unsuccessful transmission
16#C330		Error at UDP_SOCKET function block.
	16#xxxx	Refer to appendix.
16#C331		Error at UDP_RECEIVE function block.
	16#xxxx	Refer to appendix.
16#C332		Error at UDP_SEND function block.
	16#xxxx	Refer to appendix.
16#C410	16#0000	Timeout while initializing.
16#C414	16#0000	Timeout while sending (UDP_SEND).
16#C415	16#0000	Timeout while receiving (UDP_RECEIVE).

9.7 Startup example

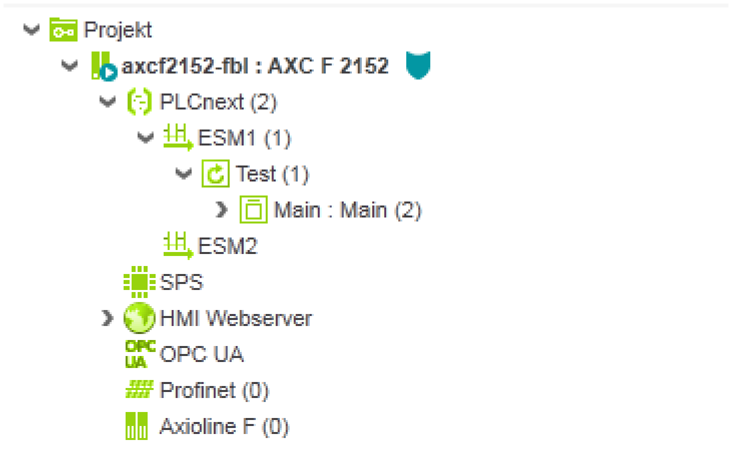
For the startup instruction please find the following example:

- ITL_8_EXA_ITL_SMTP_Client.pcwex

Plant

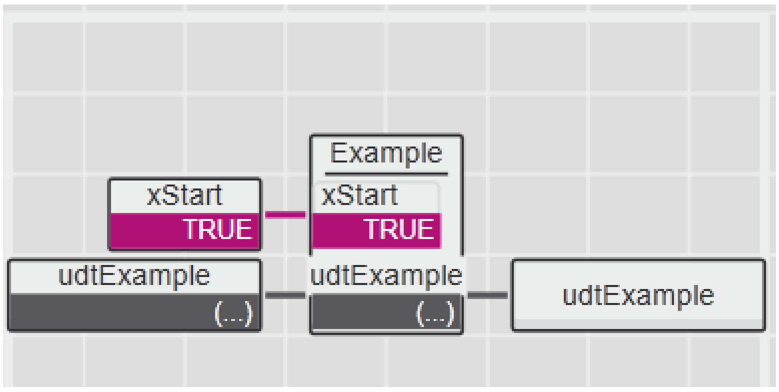
For this example, the following hardware is used:

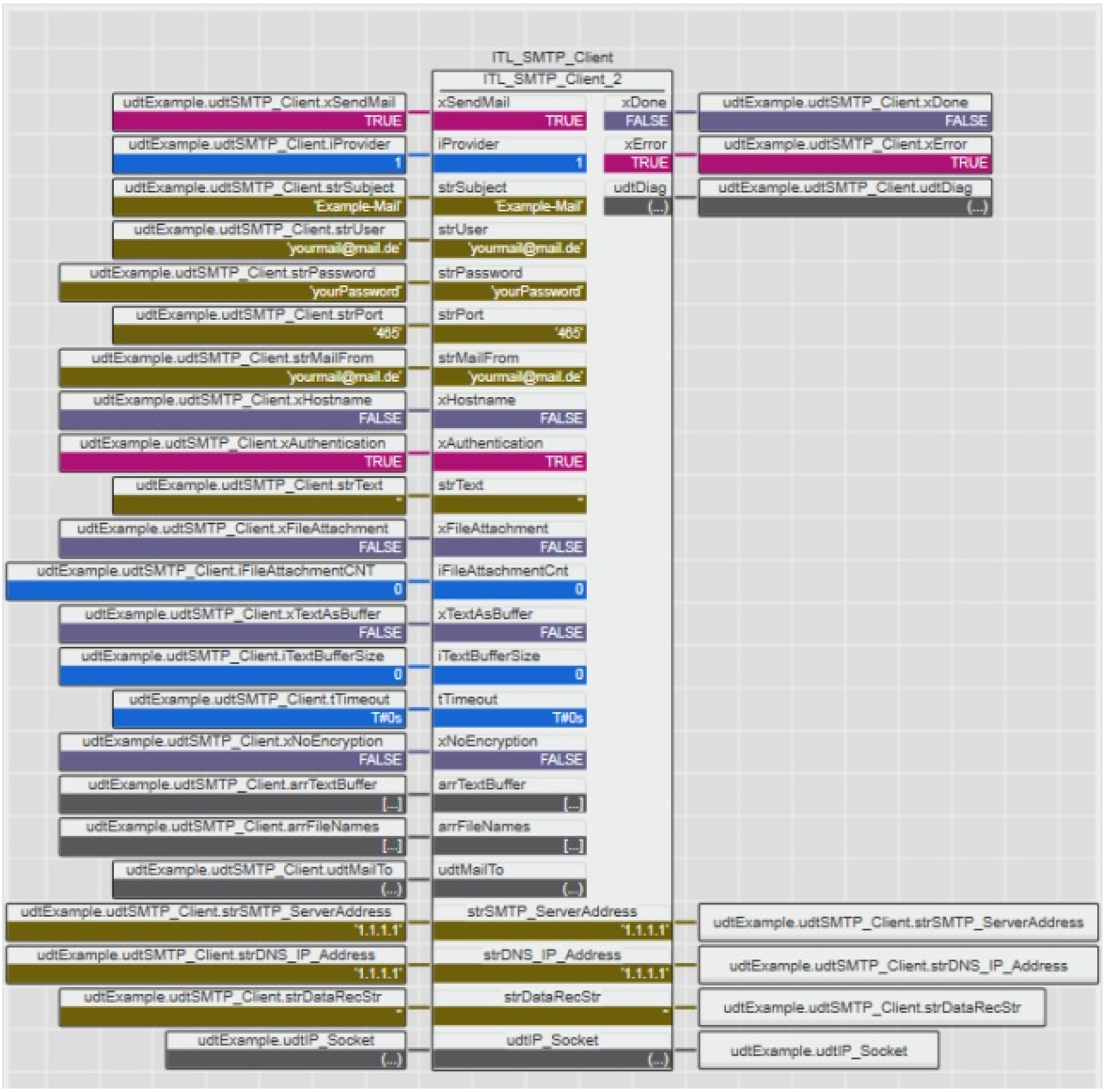
- AXC F 2152 (2404267)

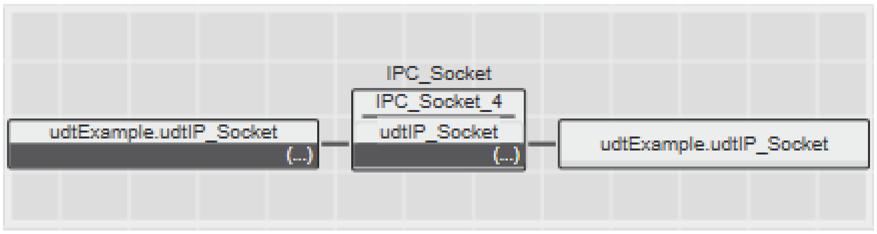


This project shows one example for the startup of ITL_SNTP_Client function block. It can be found inside the Example function block. There are state machines for every step we have to take care of when using the functionality.

To start the example set the xStart input of the Example function block to TRUE.




ITL_SMTP_Client
ITL_SMTP_Client_2
udtExample.udtSMTP_Client.xSendMail TRUE — xSendMail TRUE
udtExample.udtSMTP_Client.iProvider 1 — iProvider 1
udtExample.udtSMTP_Client.strSubject 'Example-Mail' — strSubject 'Example-Mail'
udtExample.udtSMTP_Client.strUser 'yourmail@mail.de' — strUser 'yourmail@mail.de'
udtExample.udtSMTP_Client.strPassword 'yourPassword' — strPassword 'yourPassword'
udtExample.udtSMTP_Client.strPort '465' — strPort '465'
udtExample.udtSMTP_Client.strMailFrom 'yourmail@mail.de' — strMailFrom 'yourmail@mail.de'
udtExample.udtSMTP_Client.xHostname FALSE — xHostname FALSE
udtExample.udtSMTP_Client.xAuthentication TRUE — xAuthentication TRUE
udtExample.udtSMTP_Client.strText '' — strText ''
udtExample.udtSMTP_Client.xFileAttachment FALSE — xFileAttachment FALSE
udtExample.udtSMTP_Client.iFileAttachmentCNT 0 — iFileAttachmentCnt 0
udtExample.udtSMTP_Client.xTextAsBuffer FALSE — xTextAsBuffer FALSE
udtExample.udtSMTP_Client.iTextBufferSize 0 — iTextBufferSize 0
udtExample.udtSMTP_Client.tTimeout T#0s — tTimeout T#0s
udtExample.udtSMTP_Client.xNoEncryption FALSE — xNoEncryption FALSE
udtExample.udtSMTP_Client.arrTextBuffer [...] — arrTextBuffer [...]
udtExample.udtSMTP_Client.arrFileNames [...] — arrFileNames [...]
udtExample.udtSMTP_Client.udtMailTo (...) — udtMailTo (...)
udtExample.udtSMTP_Client.strSMTP_ServerAddress '1.1.1.1' — strSMTP_ServerAddress '1.1.1.1' — udtExample.udtSMTP_Client.strSMTP_ServerAddress
udtExample.udtSMTP_Client.strDNS_IP_Address '1.1.1.1' — strDNS_IP_Address '1.1.1.1' — udtExample.udtSMTP_Client.strDNS_IP_Address
udtExample.udtSMTP_Client.strDataRecStr '' — strDataRecStr '' — udtExample.udtSMTP_Client.strDataRecStr
udtExample.udtIP_Socket (...) — udtIP_Socket (...) — udtExample.udtIP_Socket
xDone FALSE — udtExample.udtSMTP_Client.xDone FALSE
xError TRUE — udtExample.udtSMTP_Client.xError TRUE
udtDiag (...) — udtExample.udtSMTP_Client.udtDiag (...)


IPC_Socket
IPC_Socket_4
udtExample.udtIP_Socket (...) — udtIP_Socket (...) — udtExample.udtIP_Socket

Example:

Note: Please replace data with yout own valid data!

```
CASE iState OF
    0: (*wait for start of the example*)
        IF xStart = TRUE THEN
            iState  := 10;
        END_IF;
    10: (*initialization of the ITL_SMTP_Client*)
    (*set target mail adresses - replace the addresses with valid addresses*)
        udtExample.udtSMTP_Client.udtMailTO.AddressCnt      :=  2;
        udtExample.udtSMTP_Client.udtMailTO.AddressList[1]  :=  'Dummy1@mail.de';
        udtExample.udtSMTP_Client.udtMailTO.AddressList[2]  :=  'Dummy2@mail.de';
    (*replace with a valid DNS-Server address*)
        udtExample.udtSMTP_Client.strDNS_IP_Address     :=  '1.1.1.1';
        udtExample.udtSMTP_Client.xTextAsBuffer         :=  FALSE;
        udtExample.udtSMTP_Client.strSubject            :=  'Example-Mail';
        udtExample.udtSMTP_Client.xAuthentication       :=  TRUE;
        iState  := 20;
    20: (*set data for the provider*)
    (*replace data with yout own valid data*)
        udtExample.udtSMTP_Client.xHostname             :=  FALSE;
        udtExample.udtSMTP_Client.strSMTP_ServerAddress :=  '1.1.1.1';
        udtExample.udtSMTP_Client.iProvider     :=  1;
        udtExample.udtSMTP_Client.strPort       :=  '465';
        udtExample.udtSMTP_Client.strUser       :=  'yourmail@mail.de';
        udtExample.udtSMTP_Client.strMailFrom   :=  udtExample.udtSMTP_Client.strUse
        udtExample.udtSMTP_Client.strPassword   :=  'yourPassword';
        iState  := 30;
    30: (*send Mail *)
        udtExample.udtSMTP_Client.xSendMail := TRUE;
        IF udtExample.udtSMTP_Client.xDone = TRUE THEN
            iState  := 40; (*sending successfully completed*)
        END_IF;
    40: (*wait for end of Example*)
        IF xStart = FALSE THEN
            iState  := 0;
            udtExample.udtSMTP_Client.xSendMail := FALSE;
        END_IF;
END_CASE;
```

9.8 Data types

```
TYPE
    (*Buffer of 1026 byte*)
    ITL_SMTP_ARR_B_1_1026       :   ARRAY [1..1026] OF BYTE;
    (*Array for Email Address blind copy reciver*)
    ITL_SMTP_ARR_B_1_128        :   ARRAY [1..128] OF BYTE;
    (*Array for body text and diagnostic from SMTP server*)
    ITL_SMTP_STR_200            :   STRING (200);
    (*Buffer of 1460 STRINGS*)
    ITL_SMTP_STR_1460           :   STRING (1460);
    (*String for strPort or strTimeZone - max. 5 character*)
    ITL_SMTP_STR_5              :   STRING (5);
    (*Array of 8000 bit*)
    ITL_SMTP_ARR_X_1_8000       :   ARRAY [1..8000] OF BOOL;
    (*Array of BOOL for ByteBitConverter*)
    ITL_SMTP_ARR_X_0_7          :   ARRAY [0..7] OF BOOL;
    (*Array for filenames*)
    ITL_SMTP_ARR_STR_1_10       :   ARRAY [1..10] OF STRING;
    (*Array of INT for smtp codes*)
    ITL_SMTP_ARR_B_1_8          :   ARRAY [1..8] OF INT;
    (*Array of Byte for Text Buffer*)
    ITL_SMTP_ARR_B_1_3200           :   ARRAY [1..3200] OF BYTE;

    ITL_SMTP_ARR_STR_1_10_AddressList   :   ARRAY [1..10] OF STRING;
    (*struct for mail adress*)
    ITL_SMTP_UDT_EmailAddress       :   STRUCT
            AddressCnt      :   INT; (*number of addresses*)
            (*list of addresses*)
            AddressList     :   ITL_SMTP_ARR_STR_1_10_AddressList;
    END_STRUCT;

    ITL_SMTP_UDT_DIAG       :   STRUCT              (*Diag struct*)
        wDiagCode       :   WORD;
        wAddDiagCode    :   WORD;
        iState          :   INT; (*state of the internal state machine*)
        iErrorState     :   INT; (*state of the last error*)
    END_STRUCT;
END_TYPE
```

10 Appendix

10.1 TCP/UDP/TLS_*

ERROR = FALSE

Status code	Description
16#0000	Situation is normal (no error).
16#8000	Socket is trying to connect the partner.
16#8001	Server is listening for a client.
16#8002	Server has rejected a client because the IP address and port number do not match.
16#8003	Not all data could be sent. Remaining data will be sent in the next cycle(s).
16#8004	Not all data received: Received length < Expected length

ERROR = TRUE

Error code	Description	Error only for
16#C001	Socket creation failed.	
16#C002	IP has wrong format.	
16#C003	Memory allocation failed.	
16#C100	Unexpected error during connecting of a client to a server.	TCP/TLS_SOCKET
16#C101	Unexpected error during receive operation.	UDP/TCP/TLS_RECEIVE
16#C102	Unexpected error during send operation.	UDP/TCP/TLS_SEND
16#C103	Unexpected error during bind operation.	UDP_SOCKET
16#C104	Unexpected error during listen operation.	TCP/TLS_SOCKET
16#C105	Unexpected error during accept operation.	TCP/TLS_SOCKET
16#C150	The TLS parameterization of the TLS_SEND/TLS_RECEIVE function blocks is inconsistent with the TLS_SOCKET function block. This is the case when: • TLS_SEND/TLS_RECEIVE require secure transmission/reception of data (SEND_SECURE/RECEIVE_SECURE input = TRUE), but the socket is not yet initialized for TLS communication (START_TLS input of TLS_SOCKET is FALSE). • TLS_SEND/TLS_RECEIVE require insecure transmission/reception of data (SEND_SECURE/RECEIVE_SECURE input = FALSE), but the socket is already initialized for TLS communication (START_TLS input of TLS_SOCKET is TRUE).	TLS_*
16#C151	An error regarding the START_TLS input of the TLS_SOCKET function block has occurred. START_TLS was set from TRUE to FALSE during opened TLS socket (ACTIVE input = TRUE). This is the case when:	TLS_*
16#C201	There are too many open sockets in the underlying socket provider.	
16#C202	An operation on a nonblocking socket cannot be completed immediately.	
16#C204	The datagram is too long.	

16#C205	Only one use of an address is normally permitted.	
16#C206	The selected IP address is not valid in this context.	
16#C207	The connection was aborted by the .NET Framework or the underlying socket provider.	
16#C208	The connection was reset by the remote peer.	
16#C210	The application tried to send or receive data, and the Socket is not connected (_SOCKET.ACTIVE == False).	
16#C211	No such host is known. The name is not an official host name or alias.	
16#C212	An unspecified System.Net.Sockets. Socket error has occurred.	
16#C213	The remote host is actively refusing a connection.	
16#C214	An invalid argument was supplied to a System.Net.Sockets.Socket member.	
16#C215	A blocking operation is in progress.	
16#C216	The overlapped operation was aborted due to the closure of the System.Net.Sockets.Socket.	
16#C217	The application has initiated an overlapped operation that cannot be completed immediately.	
16#C218	A blocking System.Net.Sockets.Socket call was canceled.	
16#C219	An attempt was made to access a System.Net.Sockets.Socket in a way that is forbidden by its access permissions.	
16#C21A	An invalid pointer address was detected by the underlying socket provider.	
16#C21B	A System.Net.Sockets.Socket operation was attempted on a non-socket.	
16#C21C	A required address was omitted from an operation on a System.Net.Sockets.Socket.	
16#C21D	An unknown, invalid, or unsupported option or level was used with a System.Net.Sockets.Socket.	
16#C21E	The protocol type is incorrect for this System.Net.Sockets.Socket.	
16#C21F	The protocol is not implemented or has not been configured.	
16#C220	The support for the specified socket type does not exist in this address family.	
16#C221	The address family is not supported by the protocol family.	
16#C222	The protocol family is not implemented or has not been configured.	
16#C223	he address family specified is not supported. This error is returned if the IPv6 address family was specified and the IPv6 stack is not installed on the local machine. This error is returned if the IPv4 address family was specified and the IPv4 stack is not installed on the local machine.	
16#C224	The network is not available.	
16#C225	No route to the remote host exists.	
16#C226	The application tried to set System.Net.Sockets.SocketOptionName. KeepAlive on a connection that has already timed out.	
16#C227	No free buffer space is available for a System.Net.Sockets.Socket operation.	
16#C228	A request to send or receive data was disallowed because the System.Net.Sockets.Socket has already been closed.	
16#C229	The connection attempt timed out, or the connected host has failed to respond.	
16#C22A	The operation failed because the remote host is down.	

16#C22B	There is no network route to the specified host. Could not connect to DEST_IP.	
16#C22C	Too many processes are using the underlying socket provider.	
16#C22D	The network subsystem is unavailable.	
16#C22E	The version of the underlying socket provider is out of range.	
16#C22F	The underlying socket provider has not been initialized.	
16#C230	A graceful shutdown is in progress.	
16#C231	The specified class was not found.	
16#C232	The name of the host could not be resolved. Try again later.	
16#C233	The error is unrecoverable or the requested database cannot be located.	
16#C234	The requested name or IP address was not found on the name server.	

10.2 STRING_TO_BUF

Status number	Description
0	The copy process has been finished correctly.
1	The VAR_IN_OUT descriptors used for the parameter SRC and BUFFER are invalid. This is an internal error.
2	The length of the source buffer does not fit. The size of bytes to be copied assigned in BUF_CNT is larger than the available size of the SRC.
3	The length of the destination buffer does not fit. The sum of the bytes to be copied assigned in BUF_CNT and the offset in the connected byte stream assigned in BUF_OFFS is larger than the size of the connected byte stream.
4	This data type is not supported.
5	The alignment does not fit to this data type. The size to be copied assigned in BUF_CNT must be divisible without remainder by the size of the data type.
6	The conversion INTEL/MOTOROLA has failed.
7	The string length does not fit. Additional checks are necessary for the data type string. This is described in the chapter 'String specialties'.
8	The destination buffer has a wrong data type. In some cases the data type is checked. This is described in the special chapter for each data type.
9	The offset value is not correct. In some cases the offset is checked. This is described in the special chapter for each data type.
10	The BUF_CNT does not fit. In some cases the size to be copied is checked. This is described in the special chapter for each data type.
11	The addresses of the source and the destination are the same.

11 Support

For technical support please contact your local PHOENIX CONTACT agency

at https://www.phoenixcontact.com

Owner:

PHOENIX CONTACT Electronics GmbH
Business Unit Automation Systems
System Services
Library Services