

ComLib Library

TXV 003 51.02
5th edition
August 2010
subject to alterations

Changes history

Date	Edition	Change description
March 2009	1	First edition
September 2009	2	Information for ComLib_v1.3 added
November 2009	3	Description of fbRecvFrom and fbSendTo according to ComLib_v1.3 added (input variable channel was renamed to chanCode)
March 2010	4	Description of GetChanStat() function and Tuni_STAT data type Description of GetChanSettings() and SetChanSettings() functions Correspond to ComLib_v14
August 2010	5	Example of using of constants for modem control parameter of serial communication channel repaired (<i>chanSettings.modemControl</i>) Correspond to ComLib_v14

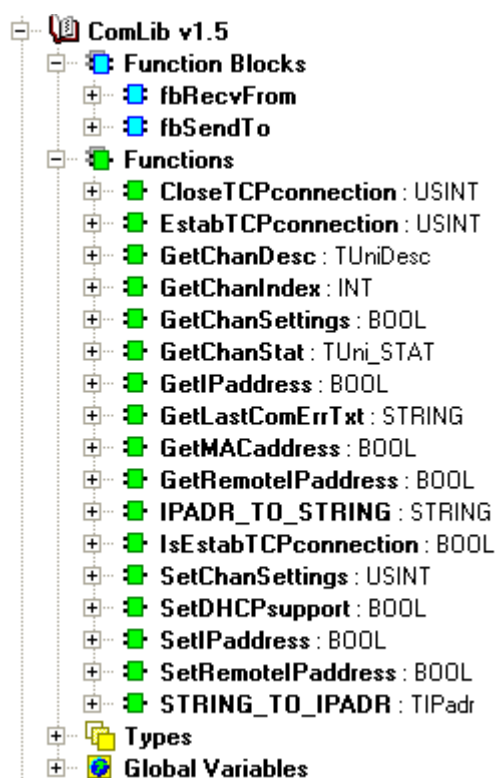
CONTENT

1 INTRODUCTION.....	4
2 DATA TYPES.....	5
2.1 Type TEthStat.....	6
2.2 Type TIPadr.....	7
2.3 Type TLocalEthAdr.....	8
2.4 Type TRemoteEthAdr.....	9
2.5 Type TMacAdr.....	10
2.6 Type TUniDesc.....	11
2.7 Type TUni_STAT.....	12
2.8 Type TChanSettings.....	13
3 CONSTANTS.....	14
4 GLOBAL VARIABLES.....	19
5 FUNCTIONS.....	20
5.1 Function GetLastComErrTxt.....	22
5.2 Function EstabTCPconnection.....	23
5.3 Function CloseTCPconnection.....	25
5.4 Function IsEstabTCPconnection.....	26
5.5 Function SetRemoteIPAddress.....	27
5.6 Function GetRemoteIPAddress.....	29
5.7 Function SetIPAddress.....	30
5.8 Function GetIPAddress.....	32
5.9 Function GetMACaddress.....	34
5.10 Function SetDHCPsupport.....	36
5.11 Function STRING_TO_IPADR.....	38
5.12 Function IPADR_TO_STRING.....	40
6 FUNCTION BLOCKS.....	42
6.1 Function block fbRecvFrom.....	43
6.2 Function block fbSendTo.....	49

1 INTRODUCTION

The ComLib library is supplied standardly as a part of Mosaic programmable environment. The library contains functions and function blocks enabling reception and sending of messages via the communication PLC channel. The channel can be either a serial line or Ethernet.

The following picture shows the structure of the ComLib library within Mosaic



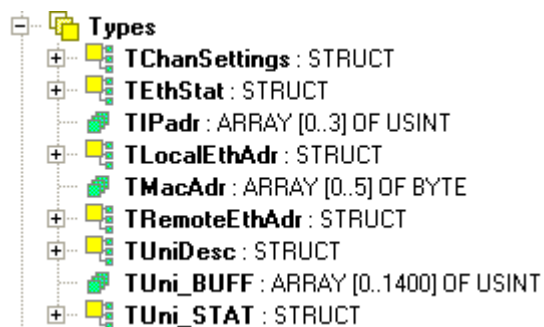
If we want to use functions from the ComLib library in the application program of the PLC, firstly, it is necessary to add this library to the project. The library is supplied as a part of installation of the Mosaic environment from version v2.0.15.0.

The ComLib library is not supported within systems TC-650, within the system TC700 is not possible to use the library with processor modules CP-7002, CP-7003 and CP-7005.

Function and function blocks of the ComLib library are supported in central units of K rank (TC700 CP-7000 and CP-7004, all versions of Foxtrot system) from version v4.4. Some functions require central units firmware v4.9 or higher (see e.g. Function SetDHCPsupport).

2 DATA TYPES

Data types in the ComLib library can be divided into two groups. First group consists of data types used internally within the library, second group consists of data types designed for use within the application program.



The brief description of data types is given in the following table:

<i>Identifier</i>	<i>Type</i>	<i>Signification</i>
<i>TEthStat</i>	STRUCT	Status information about ethernet interface
<i>TIPAdr</i>	ARRAY	IP address
<i>TLocalEthAdr</i>	STRUCT	Structure containing PLC IP address, network mask and gateway address
<i>TRemoteEthAdr</i>	STRUCT	Structure containing remote IP address, remote port and local port
<i>TmacAdr</i>	ARRAY	Array of 6 bytes designed for MAC address saving
<i>TUniDesc</i>	STRUCT	Description of communication channel mapping (used only for internal purposes)
<i>TUni_BUFF</i>	ARRAY	buffer for broadcasted or received data (used only for internal purposes)
<i>TUni_STAT</i>	STRUCT	Communication channel status zone
<i>TChanSettings</i>	STRUCT	Serial communication channel settings

2.1 Type *TEthStat*

Library : *ComLib*

```

TEthStat : STRUCT
  chan_present : BOOL
  DHCP_enabled : BOOL
  IP_obtained : BOOL
  IP_expired : BOOL
  reserved : USINT
  trueMes : UDINT
  falseMes : UDINT

```

Data type *TEthStat* is a structure containing information about the ethernet interface. This structure have global variables *ETH1_STAT* and *ETH2_STAT* that contain information about interfaces ETH1 and ETH2.

Particular items of the structure *TEhtStat* have the following signification:

<i>Identifier</i>	<i>Type</i>	<i>Signification</i>
<i>TEthStat</i>	STRUCT	Structure containing information about Ethernet interface
<i>.chan_present</i>	BOOL	Ethernet channel is present (is mounted)
<i>.DHCP_enabled</i>	BOOL	Automatic assignment of IP address by DHCP server enabled
<i>.IP_obtained</i>	BOOL	IP address assigned by DHCP server
<i>.IP_expired</i>	BOOL	Validity of automatically assigned IP address expired
<i>.reserved</i>	USINT	Reserved for further use
<i>.trueMes</i>	UDINT	Total number of packets that were processed by the system
<i>.falseMes</i>	UDINT	Number of packets their processing was denied (the reason can be invalid packet or a packet with protocol that is not supported by the control system)

See also Global variables

2.2 Type *TIPadr*

·  **TIPadr** : ARRAY [0..3] OF USINT

Library : *ComLib*

Data type *TIPadr* is an array of 4 items of USINT type. It is used in cases when it is necessary to define the IP address or network mask.

Type *TIPadr* is in the library declared as follows:

```
TYPE
  TIPadr : ARRAY [0..3] OF USINT;
END_TYPE
```

The item with index 0 contains the first number of IP address, the item with index 3 contains the last number of IP address.

The example of usage of data type *TIPadr*:

```
VAR
  my_IP_adr    : TIPadr := [192, 168, 001, 010]; // 192.168.1.10
  my_net_mask : TIPadr := [255, 255, 255, 000]; // 255.255.255.0
END_VAR
```

See also Type *TLocalEthAdr*, Type *TRemoteEthAdr*

2.3 Type *TLocalEthAdr*

Library : *ComLib*

The structure of *TLocalEthAdr* type is used by functions *GetIPAddress* and *SetIPAddress* to transfer the IP address, mask and gate address. All items of this structure are of *TIPadr* type which is the array with four items of USINT type.

Significance of particular items of the structure *TLocalEthAdr* is as follows:

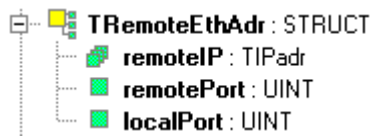
- *IP* IP address (e.g. 192. 168.001.010)
- *IM* IP mask (e.g. 255.255.255.0)
- *GW* gate address (e.g. 192. 168.001.100)

The example of initialization of the variable of *TLocalEthAdr* type:

```
VAR
  new_eth_adr : TLocalEthAdr := ( IP:= [192,168,001,010],
                                   IM:= [255,255,255,000],
                                   GW:= [192,168,001,100] );
END_VAR
```

See also Function *GetIPAddress*, Function *SetIPAddress*

2.4 Type *TRemoteEthAdr*

Library : *ComLib*

The structure of *TRemoteEthAdr* type uses function *SetRemoteIPAddress* to set the target IP address, target port and local port. The function can be used for ethernet channels ETH1 and ETH2 that are set to universal mode with switched on protocol TCP or UDP. Channel ETH1 is generally located in the PLC processor unit, channel ETH2 can be found on the communication unit (e.g. SC-7102, etc.).

Signification of particular items of the structure *TRemoteEthAdr* is as follows:

- *remoteIP* target IP address (e.g. 192. 168.001.010)
- *remotePort* target port (i.e. port where the message by TCP protocol is sent to, e.g. 61000)
- *localPort* source port (i.e. port where the message by TCP protocol is sent from, e.g. 61001)

The example of initialization of the variable of *TremoteEthAdr* type:

```
VAR
  new_eth_adr : TRemoteEthAdr := ( remoteIP   := [192,168,001,010],
                                   remotePort := 61000,
                                   localPort  := 61001 );
END_VAR
```

See also Function *GetRemoteIPAddress*, Function *SetRemoteIPAddress*

2.5 Type *TMacAdr*

Library : *ComLib* **TMacAdr** : ARRAY [0..5] OF BYTE

The array of *TMacAdr* type uses function *GetMacAddress* that detects MAC address of the ethernet interface.

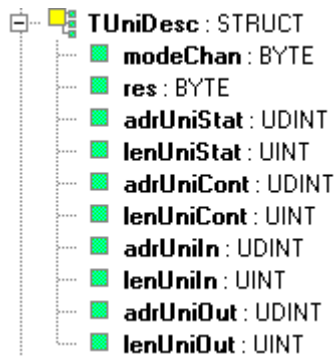
The example of the usage of the variable of *TmacAdr* type:

```
PROGRAM prgTestGetMAC
VAR
  mac_adr   : TMacAdr;
  tmp       : BOOL;
  message    : STRING;
END_VAR

tmp := GetMACaddress( EthChan := ETH1, MacAdr := my_mac_adr);
IF (mac_adr[0] = 0) AND (mac_adr[1] = 16#0A) AND (mac_adr[2] = 16#14) THEN
  message := 'This is Teco device';
END_IF;
END_PROGRAM
```

See also Function *GetMACaddress*

2.6 Type *TUniDesc*

Library : *ComLib*

Data type *TUniDesc* is a structure describing communication channel mapping that is returned by function *GetChanDesc()*. This is a service function used for internal purposes of the library.

Type *TUniDesc* is in the library declared as follows:

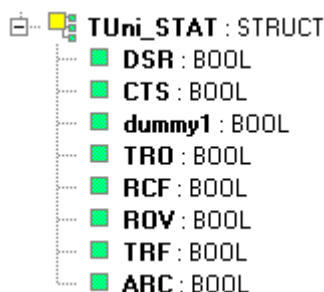
```

TYPE
  TUniDesc : STRUCT
    modeChan  : byte;           // channel mode
    res       : byte;           // reserve
    adrUniStat : uint;          // status zone address
    lenUniStat : uint;          // status zone length
    adrUniCont : uint;          // control zone address
    lenUniCont : uint;          // control zone length
    adrUniIn   : uint;          // receiving zone address
    lenUniIn   : uint;          // receiving zone length
    adrUniOut  : uint;          // sending zone address
    lenUniOut  : uint;          // sending zone length
  END_STRUCT;
END_TYPE
  
```

Channel that is set within the universal mode sets the item *modeChan* in the structure *TUniDesc* to the value 5.

See also Type *TLocalEthAdr*, Type *TRemoteEthAdr*

2.7 Type *TUni_STAT*

Library : *ComLib*

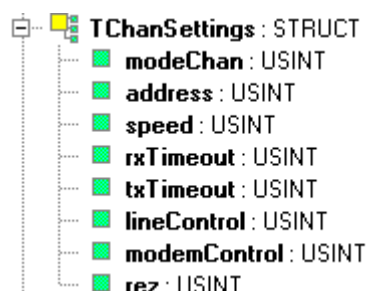
Data type *TUni_STAT* is a structure describing actual status of the communication channel that is returned by function *GetChanStat()*.

Signification of particular items of the structure *TUni_STAT* is as follows:

Identifier	Type	Signification
<i>TUni_STAT</i>	STRUCT	Structure containing information on the communication channel actual status
<i>.DSR</i>	BOOL	DSR signal status
<i>.CTS</i>	BOOL	CTS signal status (ready to sending)
<i>.dummy1</i>	BOOL	Reserved for the next use
<i>.TRO</i>	BOOL	Sending stacks are overfill, next message entry will be invalid (log.1)
<i>.RCF</i>	BOOL	Receiving stacks are overfill, actually received message will be lost (log.1)
<i>.ROV</i>	BOOL	overflow (log.1) – received message is longer than reserved receiving zone
<i>.TRF</i>	BOOL	Sending in progress, next message entry will be accepted after transmitting (log.1)
<i>.ARC</i>	BOOL	Receiving alternation- when new message is received the bit is changed

See also Chyba: zdroj odkazu nenalezen

2.8 Type *TChanSettings*

Library : *ComLib*

Data type *TChanSettings* is a structure containing an actual settings of a serial communication channel that is set by function *GetChanSettings()*. The same structure is used by function *SetChanSettings()* that can be used for changing of serial channel settings.

Signification of particular items of the structure *TchanSettings* is as follows

Identifier	Type	Signification
<i>TChanSettings</i>	STRUCT	Structure containing information on the communication channel actual settings
<i>.modeChan</i>	USINT	Communication channel mode see constants <i>MODE_OFF</i> , <i>MODE_PC</i> , <i>MODE_UNI</i> ,...
<i>.address</i>	USINT	Address for communication
<i>.speed</i>	USINT	Communication speed see constants <i>BAUD_50</i> , <i>BAUD_100</i> ,...
<i>.rxTimeout</i>	USINT	Timeout for receiving (minimal idle time for the line that corresponds to the number of received bytes after that receiving of the next message will begin). After elapsing this time the received message is thought to complete.
<i>.txTimeout</i>	USINT	Timeout for sending (minimal guaranteed idle time between two sent messages corresponding to number of sent bytes). This parameter provides that the idle time at minimum of this length will be kept on the line between two sent messages.
<i>.lineControl</i>	USINT	Parity, number of bits, number of stopbits see also constants <i>PARITY_ODD</i> , <i>PARITY_EVEN</i> , ...
<i>.modemControl</i>	USINT	Modem signals (RTS signal settings) see also constants <i>RTS_AUTO</i> , <i>HALF_DUPLEX</i> ,...
<i>.rez</i>	USINT	reserve

See also Chyba: zdroj odkazu nenalezen, Chyba: zdroj odkazu nenalezen

3 CONSTANTS

There are following constants defined within the ComLib library:

```

VAR_GLOBAL CONSTANT
  ■ MODE_OFF : USINT := 16#00
  ■ MODE_PC : USINT := 16#02
  ■ MODE_UNI : USINT := 16#05
  ■ MODE_MPC : USINT := 16#06
  ■ MODE_MDB : USINT := 16#07
  ■ MODE_PFB : USINT := 16#08
  ■ BAUD_50 : USINT := 16#01
  ■ BAUD_100 : USINT := 16#02
  ■ BAUD_200 : USINT := 16#03
  ■ BAUD_300 : USINT := 16#04
  ■ BAUD_600 : USINT := 16#05
  ■ BAUD_1200 : USINT := 16#06
  ■ BAUD_2400 : USINT := 16#07
  ■ BAUD_4800 : USINT := 16#08
  ■ BAUD_9600 : USINT := 16#0A
  ■ BAUD_14400 : USINT := 16#0B
  ■ BAUD_19200 : USINT := 16#0C
  ■ BAUD_28800 : USINT := 16#0D
  ■ BAUD_38400 : USINT := 16#0E
  ■ BAUD_57600 : USINT := 16#10
  ■ BAUD_76800 : USINT := 16#12
  ■ BAUD_93750 : USINT := 16#13
  ■ BAUD_115200 : USINT := 16#14
  ■ NO_PARITY : USINT := 16#00
  ■ PARITY_ODD : USINT := 16#08
  ■ PARITY_EVEN : USINT := 16#18
  ■ PARITY_0 : USINT := 16#28
  ■ PARITY_1 : USINT := 16#38
  ■ SEVEN_BITS : USINT := 16#40
  ■ EIGHT_BITS : USINT := 16#00
  ■ ONE_STOP_BIT : USINT := 16#00
  ■ TWO_STOP_BITS : USINT := 16#80
  ■ RTS_0 : USINT := 16#00
  ■ RTS_1 : USINT := 16#02
  ■ RTS_MAN : USINT := 16#40
  ■ RTS_AUTO : USINT := 16#80
  ■ RTS_CTS_AUTO : USINT := 16#C0
  ■ HALF_DUPLEX : USINT := 16#08
  ■ ANY_IP : TIPadr := [0]
  ■ ETH1_uni0 : UINT := 16#07E1
  ■ ETH1_uni1 : UINT := 16#17E1
  ■ ETH1_uni2 : UINT := 16#27E1
  ■ ETH1_uni3 : UINT := 16#37E1
  ■ ETH1_uni4 : UINT := 16#47E1
  ■ ETH1_uni5 : UINT := 16#57E1
  ■ ETH1_uni6 : UINT := 16#67E1
  ■ ETH1_uni7 : UINT := 16#77E1
  ■ ETH2_uni0 : UINT := 16#07E2
  ■ ETH2_uni1 : UINT := 16#17E2
  ■ ETH2_uni2 : UINT := 16#27E2
  ■ ETH2_uni3 : UINT := 16#37E2
  ■ ETH2_uni4 : UINT := 16#47E2
  ■ ETH2_uni5 : UINT := 16#57E2
  ■ ETH2_uni6 : UINT := 16#67E2
  ■ ETH2_uni7 : UINT := 16#77E2
  ■ CH1_uni : UINT := 16#0101
  ■ CH2_uni : UINT := 16#0202
  ■ CH3_uni : UINT := 16#0103
  ■ CH4_uni : UINT := 16#0204
  ■ CH5_uni : UINT := 16#0105
  ■ CH6_uni : UINT := 16#0206
  ■ CH7_uni : UINT := 16#0107
  ■ CH8_uni : UINT := 16#0208
  ■ CH9_uni : UINT := 16#0109
  ■ CH10_uni : UINT := 16#020A
  ■ ETH1 : USINT := 16#E1
  ■ ETH2 : USINT := 16#E2
  ■ ETH3 : USINT := 16#E3
  ■ ETH4 : USINT := 16#E4
  ■ SCH1 : USINT := 16#01
  ■ SCH2 : USINT := 16#02
  ■ SCH3 : USINT := 16#03
  ■ SCH4 : USINT := 16#04
  ■ SCH5 : USINT := 16#05
  ■ SCH6 : USINT := 16#06
  ■ SCH7 : USINT := 16#07
  ■ SCH8 : USINT := 16#08
  ■ SCH9 : USINT := 16#09
  ■ SCH10 : USINT := 16#0A
  ■ COM_OK : USINT := 0
  ■ COM_ERR1 : USINT := 1
  ■ COM_ERR2 : USINT := 2
  ■ COM_ERR3 : USINT := 3
  ■ COM_ERR4 : USINT := 4
  ■ COM_ERR5 : USINT := 5
  ■ COM_ERR6 : USINT := 6
  ■ COM_ERR7 : USINT := 7
  ■ COM_ERR8 : USINT := 8
  ■ COM_ERR16 : USINT := 16#10
  ■ COM_ERR17 : USINT := 16#11
  ■ COM_ERR18 : USINT := 16#12
  ■ COM_ERR19 : USINT := 16#13
  ■ COM_ERR20 : USINT := 16#14
  ■ COM_ERR24 : USINT := 16#18
  ■ COM_ERR25 : USINT := 16#19
  ■ COM_ERR49 : USINT := 16#31
  ■ COM_ERR50 : USINT := 16#32
  ■ COM_ERR64 : USINT := 16#40

```

Constants *ETH1_uni0* up to *ETH1_uni7* are used for specification of Ethernet channel within functions *EtabTCPconnection*, *CloseTCPconnection*, *SetRemoteIPaddress* and in function blocks *fbSendTo* and *fbRecvFrom*. The same meaning have constants *ETH2_uni0* up to *ETH2_uni7*.

Identifier	Type	Value	Signification
<i>ETH1_uni0</i>	UINT	16#07E1	ethernet channel ETH1, connection uni0
<i>ETH1_uni1</i>	UINT	16#17E1	ethernet channel ETH1, connection uni1
<i>ETH1_uni2</i>	UINT	16#27E1	ethernet channel ETH1, connection uni2
<i>ETH1_uni3</i>	UINT	16#37E1	ethernet channel ETH1, connection uni3
<i>ETH1_uni4</i>	UINT	16#47E1	ethernet channel ETH1, connection uni4
<i>ETH1_uni5</i>	UINT	16#57E1	ethernet channel ETH1, connection uni5
<i>ETH1_uni6</i>	UINT	16#67E1	ethernet channel ETH1, connection uni6
<i>ETH1_uni7</i>	UINT	16#77E1	ethernet channel ETH1, connection uni7
<i>ETH2_uni0</i>	UINT	16#07E2	ethernet channel ETH2, connection uni0
<i>ETH2_uni1</i>	UINT	16#17E2	ethernet channel ETH2, connection uni1
<i>ETH2_uni2</i>	UINT	16#27E2	ethernet channel ETH2, connection uni2
<i>ETH2_uni3</i>	UINT	16#37E2	ethernet channel ETH2, connection uni3
<i>ETH2_uni4</i>	UINT	16#47E2	ethernet channel ETH2, connection uni4
<i>ETH2_uni5</i>	UINT	16#57E2	ethernet channel ETH2, connection uni5
<i>ETH2_uni6</i>	UINT	16#67E2	ethernet channel ETH2, connection uni6
<i>ETH2_uni7</i>	UINT	16#77E2	ethernet channel ETH2, connection uni7

Similarly, constants *CH1_uni* up to *CH10_uni* determine the concrete serial channel during the function blocks *fbSendTo* and *fbRecvFrom* call.

Identifier	Type	Value	Signification
<i>CH1_uni</i>	UINT	16#0101	serial channel CH1, uni mode
<i>CH2_uni</i>	UINT	16#0202	serial channel CH2, uni mode
<i>CH3_uni</i>	UINT	16#0103	serial channel CH3, uni mode
<i>CH4_uni</i>	UINT	16#0204	serial channel CH4, uni mode
<i>CH5_uni</i>	UINT	16#0105	serial channel CH5, uni mode
<i>CH6_uni</i>	UINT	16#0206	serial channel CH6, uni mode
<i>CH7_uni</i>	UINT	16#0107	serial channel CH7, uni mode
<i>CH8_uni</i>	UINT	16#0208	serial channel CH8, uni mode
<i>CH9_uni</i>	UINT	16#0109	serial channel CH9, uni mode
<i>CH10_uni</i>	UINT	16#020A	serial channel CH10, uni mode

Constants *ETH1* up to *ETH2* are used for ethernet interface selection within functions *SetIPaddress*, *GetIPaddress* and *GetMACaddress*.

Identifier	Type	Value	Signification
<i>ETH1</i>	USINT	16#E1	ethernet interface on the PLC central unit
<i>ETH2</i>	USINT	16#E2	ethernet interface on communication module (e.g. SC-7102 for TC700 system)
<i>ETH3</i>	USINT	16#E3	ethernet interface on communication module (e.g. SC-7104 for TC700 system)
<i>ETH4</i>	USINT	16#E4	ethernet interface on communication module (e.g. SC-7104 for TC700 system)

Similarly, constants *SCH1* up to *SCH10* determine the concrete serial channel during the functions *GetChanSettings* and *SetChanSettings* call.

Identifier	Type	Value	Signification
<i>SCH1</i>	USINT	16#01	serial channel CH1
<i>SCH2</i>	USINT	16#02	serial channel CH2
<i>SCH3</i>	USINT	16#03	serial channel CH3
<i>SCH4</i>	USINT	16#04	serial channel CH4
<i>SCH5</i>	USINT	16#05	serial channel CH5
<i>SCH6</i>	USINT	16#06	serial channel CH6
<i>SCH7</i>	USINT	16#07	serial channel CH7
<i>SCH8</i>	USINT	16#08	serial channel CH8
<i>SCH9</i>	USINT	16#09	serial channel CH9
<i>SCH10</i>	USINT	16#0A	serial channel CH10

Constants *COM_OK*, *COM_ERR1* up to *COM_ERR64* are release values that return function blocks *fbSendTo* and *fbRecvFrom* in the output variable *error*. The signification of particular constants is as follows:

Identifier	Type	Value	Signification
<i>COM_OK</i>	USINT	0	Communication with no errors
<i>COM_ERR1</i>	USINT	1	Required channel is not set within the universal mode
<i>COM_ERR2</i>	USINT	2	Too many sent data or, rather, too small buffer of the channel in the uni mode
<i>COM_ERR3</i>	USINT	3	Too many received data or, rather, received data does not fit in the specified variable
<i>COM_ERR4</i>	USINT	4	Communication channel invalid number
<i>COM_ERR5</i>	USINT	5	Previous message was not sent yet (sending stacks are overfilled)

<i>COM_ERR6</i>	USINT	6	Zero length of sent data
<i>COM_ERR16</i>	USINT	16	Incorrect initial character
<i>COM_ERR17</i>	USINT	17	Parity error
<i>COM_ERR18</i>	USINT	18	Maximum message length exceeded
<i>COM_ERR19</i>	USINT	19	Incorrect second byte of the confirmation
<i>COM_ERR20</i>	USINT	20	Incorrect second byte of the terminator
<i>COM_ERR24</i>	USINT	24	Invalid check sum
<i>COM_ERR25</i>	USINT	25	Invalid terminator
<i>COM_ERR49</i>	USINT	49	Invalid length of sent data
<i>COM_ERR50</i>	USINT	50	Zero length of sent data
<i>COM_ERR64</i>	USINT	64	Timeout not adhered
<i>COM_ERRc6</i>	USINT	198	Serial channel is not in requested mode

Constants *MODE_xx* are used for the determination of serial communication channel mode (see *TchanSettings.modeChan*)

Identifier	Type	Value	Signification
<i>MODE_OFF</i>	USINT	16#01	No mode set , channel is not operated
<i>MODE_PC</i>	USINT	16#02	Mode PC
<i>MODE_UNI</i>	USINT	16#05	Mode uni
<i>MODE_MPC</i>	USINT	16#06	Mode MPC
<i>MODE_MDB</i>	USINT	16#07	Mode Modbus (slave)
<i>MODE_PFB</i>	USINT	16#08	Mode Profibus DP (master)

Constants *BAUD_xx* are used for the determination of serial communication channel speed (see *TchanSettings.speed*)

Identifier	Type	Value	Signification
<i>BAUD_50</i>	USINT	16#01	Communication speed 50 Baud
<i>BAUD_100</i>	USINT	16#02	Communication speed 100 Baud
<i>BAUD_200</i>	USINT	16#03	Communication speed 200 Baud
<i>BAUD_300</i>	USINT	16#04	Communication speed 300 Baud
<i>BAUD_600</i>	USINT	16#05	Communication speed 600 Baud
<i>BAUD_1200</i>	USINT	16#06	Communication speed 1200 Baud
<i>BAUD_2400</i>	USINT	16#07	Communication speed 2400 Baud
<i>BAUD_4800</i>	USINT	16#08	Communication speed 4800 Baud
<i>BAUD_9600</i>	USINT	16#0A	Communication speed 9600 Baud
<i>BAUD_14400</i>	USINT	16#0B	Communication speed 14400 Baud

<i>BAUD_19200</i>	USINT	16#0C	Communication speed 19200 Baud
<i>BAUD_28800</i>	USINT	16#0D	Communication speed 28800 Baud
<i>BAUD_38400</i>	USINT	16#0E	Communication speed 38400 Baud
<i>BAUD_57600</i>	USINT	16#10	Communication speed 57600 Baud
<i>BAUD_76800</i>	USINT	16#12	Communication speed 76800 Baud
<i>BAUD_93750</i>	USINT	16#13	Communication speed 93750 Baud
<i>BAUD_11500</i>	USINT	16#14	Communication speed 11500 Baud

The following constants are used for the parameter of line control of the serial communication channel (see *TchanSettings.lineControl*). The resulting settings is obtained as a logical sum of required parameters. For example *chanSettings.lineControl := EIGHT_BITS OR PARITY_EVEN OR ONE_STOP_BIT*.

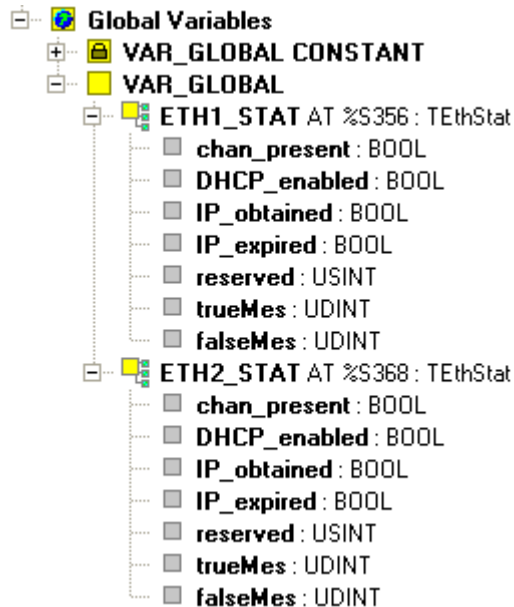
Identifier	Type	Value	Signification
<i>NO_PARITY</i>	USINT	16#00	No parity
<i>PARITY_ODD</i>	USINT	16#08	Odd parity
<i>PARITY_EVEN</i>	USINT	16#18	Even parity
<i>PARITY_0</i>	USINT	16#28	Parity permanently 0
<i>PARITY_1</i>	USINT	16#38	Parity permanently 1
<i>SEVEN_BITS</i>	USINT	16#40	Seven bits per character
<i>EIGHT_BITS</i>	USINT	16#00	Eight bits per character
<i>ONE_STOP_BIT</i>	USINT	16#00	One STOP bit
<i>TWO_STOP_BIT</i>	USINT	16#80	Two STOP bits

The constants mentioned in the following table are determined for modem control parameter of the serial communication channel (see *TchanSettings.modemControl*). The resulting settings is obtained as a logical sum of required parameters. For example *chanSettings.modemControl := RTS_AUTO OR HALF_DUPLEX*.

Identifier	Type	Value	Signification
<i>RTS_0</i>	USINT	16#00	Signal RTS permanently 0
<i>RTS_1</i>	USINT	16#02	Signal RTS permanently 1
<i>RTS_MAN</i>	USINT	16#40	Signal RTS controlled from PLC program
<i>RTS_AUTO</i>	USINT	16#80	Signal RTS set automatically (according to transmitting)
<i>RTS_CTS_AUTO</i>	USINT	16#C0	Signals RTS and CTS automatically
<i>HALF_DUPLEX</i>	USINT	16#08	Half duplex (when sending message the receiver is switched off)

4 GLOBAL VARIABLES

In the ComLib library following global variables are defined:



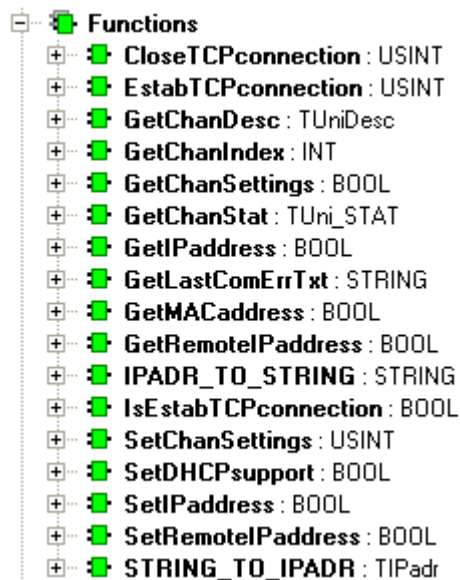
Global variables *ETH1_STAT* and *ETH2_STAT* contain information about status of ethernet interface ETH1 and ETH2. These variables are of *TEthStat* type that is also defined in the library.

Particular items of the structure *TEthStat* have the following signification:

Identifier	Type	Signification
<i>TEthStat</i>	STRUCT	Structure containing information about status of ethernet interface.
<i>.chan_present</i>	BOOL	Ethernet channel is present (is mounted)
<i>.DHCP_enabled</i>	BOOL	Automatic assignment of IP address by DHCP server enabled
<i>.IP_obtained</i>	BOOL	IP address was assigned by DHCP server
<i>.IP_expired</i>	BOOL	Validity of automatically assigned IP address expired
<i>.reserved</i>	USINT	Reserve for further use
<i>.trueMes</i>	UDINT	Total number of packets that were processed by the system
<i>.falseMes</i>	UDINT	Number of packets their processing was denied (the reason can be invalid packet or a packet with protocol that is not supported by the control system)

5 FUNCTIONS

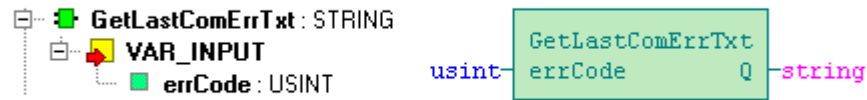
The ComLib library contains following functions:



<i>Function</i>	<i>Description</i>
<i>GetChanDesc</i>	service function, used internally within the library that returns the communication channel descriptor
<i>GetChanIndex</i>	service function, used internally within the library that returns the communication channel index
<i>GetLastComErrTxt</i>	function returns the text of an error occurred during communication
<i>EstabTCPconnection</i>	function sets up the TCP connection
<i>CloseTCPconnection</i>	function terminates TCP connection
<i>IsEstabTCPconnectio</i>	function tests if the TCP connection is set up
<i>SetRemoteIPAddress</i>	function sets the remote IP address, remote port and local port for the given ethernet channel
<i>GetRemoteIPAddress</i>	function returns actual remote IP address, remote port and local port for the given ethernet channel
<i>GetIPAddress</i>	get the actual IP address, mask and gate address
<i>SetIPAddress</i>	set new IP address, mask and gate address
<i>GetMACAddress</i>	get MAC address of the given ethernet interface
<i>SetDHCPsupport</i>	switch on the support fo automatic IP address assignment by DHCP server
<i>STRING_TO_IPADR</i>	IP address conversion from a string to the Ipadr structure
<i>IPADR_TO_STRING</i>	IP address conversion from the structure Ipadr to the string
<i>GetChanStat</i>	Function returns communication channel status

<i>Function</i>	<i>Description</i>
<i>GetChanSettings</i>	Function returns serial communication channel settings (communication speed, timeouts, etc.)
<i>SetChanSettings</i>	Function sets up parameters of serial communication channel (communication speed, timeouts, etc.)

5.1 Function GetLastComErrTxt

Library : *ComLib*

The *GetLastComErrTxt* function returns the text describing an error occurred during communication. The input parameter of the function is the error code that is returned by function blocks *fbSendTo* and *fbRecvFrom*.

This function is supported within central units of K rank (TC700 CP-7004, Foxtrot) from version v4.4.

Variable description :

	Variable	Type	Signification
VAR_INPUT			
	<i>errCode</i>	USINT	Error code indicated by function block of the type <i>fbSendTo</i> or <i>fbRecvFrom</i>
GetLastComErrTxt			
	<i>Release value</i>	STRING	Communication error description

The example of the program with the *GetLastComErrTxt* function call:

```

PROGRAM ExampleGetLastComErrTxt
VAR
  RecvFromCH1 : fbRecvFrom;
  rxBuf       : STRING[100];
  errMsg      : STRING;
END_VAR

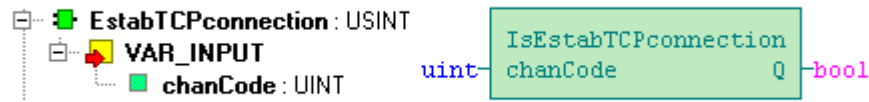
// receiving
RecvFromCH1( rq      := TRUE, chanCode := CH1_uni,
              lenRx   := 100, data      := void(rxBuf));

if RecvFromCH1.mesRec then
  // new message received
  if RecvFromCH1.error = 0 then
    // process new message
    // ...
  else
    // show error as a text
    errMsg := GetLastComErrTxt( RecvFromCH1.error);
  end if;
end if;
END_PROGRAM
  
```

See also Function block *fbRecvFrom*, Function block *fbSendTo*

5.2 Function EstabTCPconnection

Library : ComLib



The *EstabTCPconnection* function initiates the process of TCP connection setup if the previous connection is terminated. The function input parameter is the code of the communication channel (*ETH1_uni0*, ..., *ETH2_uni7*). Function returns the error code. If everything is in order, the function returns *COM_OK* (0, no error). Function is worth in case of the ethernet channel *ETH1* or *ETH2* that must be set to the mode uni – master TCP. Setting up the TCP connection is a process during which the client (PLC) and server exchange synchronization frames via TCP protocol. The period of connection establishment is dependant on many conditions (e.g. if communication is undertaken within the local network, etc.). Information whether the connection was successfully set up can be retrieved using the function *IsEstabTCPconnection*.

This function is supported on central units of K rank (TC700 CP-7004, Foxtrot) from version v4.4.

Variable description :

	Variable	Type	Signification
VAR_INPUT			
	chanCode	USINT	Communication channel selection (<i>ETH1_uni0</i> , ..., <i>ETH1_uni7</i> , <i>ETH2_uni0</i> , ..., <i>ETH2_uni7</i>)
EstabTCPconnection			
	Release value	USINT	0 if there is no error (<i>COM_OK</i>) error code in other cases (<i>COM_ERR1</i> , ..., <i>COM_ERR64</i>)

The example of the program with the *EstabTCPconnection* function call:

```

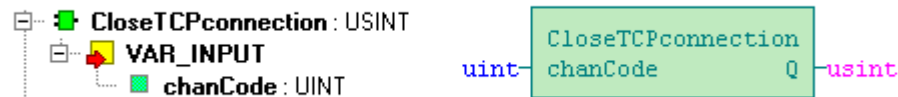
PROGRAM prgExampleEstabTCPcon
  VAR
  END_VAR

  IF NOT IsEstabTCPconnection(chanCode := ETH1_UNI0) THEN
    EstabTCPconnection(chanCode := ETH1_UNI0);
  END_IF;
END_PROGRAM

```

See also Function CloseTCPconnection, Function IsEstabTCPconnection

5.3 Function CloseTCPconnection



Library : *ComLib*

The *CloseTCPconnection* function terminates TCP connection. Function input parameter is the code of the communication channel (*ETH1_uni0*, ..., *ETH2_uni7*). Function returns error code. If everything is in order, function returns *COM_OK* (0, no error). Function is worth in case of the ethernet channel ETH1 or ETH2 that must be set to the mode uni – master TCP. Information whether the connection was successfully set up can be retrieved using the function *IsEstabTCPconnection*.

This function is supported on central units of K rank (TC700 CP-7004, Foxtrot) from version v4.4.

Variable description :

	Variable	Type	Signification
VAR_INPUT			
	<i>chanCode</i>	USINT	Communication channel selection (<i>ETH1_uni0</i> , ..., <i>ETH1_uni7</i> , <i>ETH2_uni0</i> , ..., <i>ETH2_uni7</i>)
CloseTCPconnection			
	<i>Release value</i>	USINT	0 if there is no error (<i>COM_OK</i>) error code in other cases (<i>COM_ERR1</i> , ..., <i>COM_ERR64</i>)

The example of the program with the *CloseTCPconnection* function call:

```

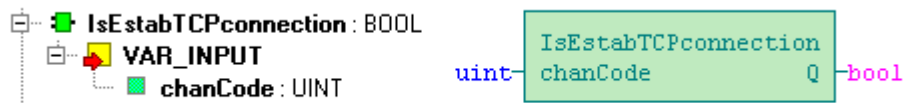
PROGRAM prgExampleIsEstabTCPcon
VAR
END_VAR

IF IsEstabTCPconnection(chanCode := ETH1_UNI0) THEN
  CloseTCPconnection(chanCode := ETH1_UNI0);
END_IF;
END_PROGRAM

```

See also Function EstabTCPconnection, Function IsEstabTCPconnection

5.4 Function *IsEstabTCPconnection*

Library : *ComLib*

The *IsEstabTCPconnection* function tests whether the TCP connection is established. Function input parameter is the code of the communication channel (*ETH1_uni0*, ..., *ETH2_uni7*). Function returns value TRUE if the connection is established, if the connection is terminated, it returns value FALSE. Communication channel must be set to the mode uni, type of the protocol TCP master or TCP slave.

This function is supported on central units of K rank (TC700 CP-7004, Foxtrot) from version v4.4.

Description variable :

	<i>Variable</i>	<i>Type</i>	<i>Signification</i>
VAR_INPUT			
	<i>chanCode</i>	USINT	Communication channel selection (ETH1_uni0, ..., ETH1_uni7, ETH2_uni0, ..., ETH2_uni7)
IsEstabTCPconnection			
	<i>Release value</i>	BOOL	TRUE if the connection is established FALSE in other cases

The example of the program with the *IsEstabTCPconnection* function call:

```

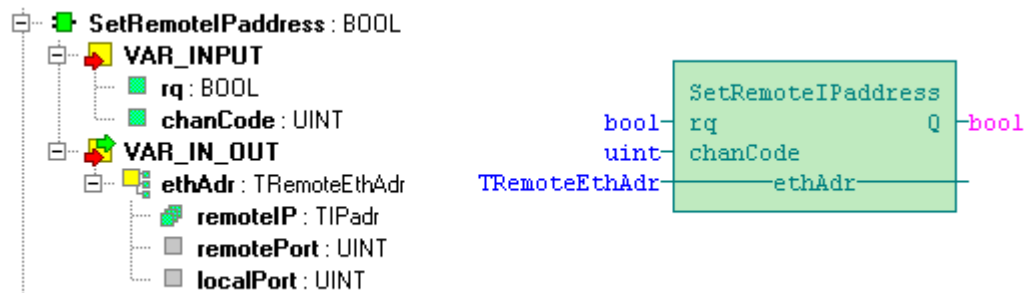
PROGRAM prgExampleIsEstabTCPcon
  VAR
  END_VAR

  IF IsEstabTCPconnection(chanCode := ETH1_UNI0) THEN
    CloseTCPconnection(chanCode := ETH1_UNI0);
  END_IF;
END_PROGRAM

```

See also Function CloseTCPconnection, Function EstabTCPconnection

5.5 Function SetRemoteIPAddress

Library : *ComLib*

The *SetRemoteIPAddress* function sets the remote IP address, number of the remote port and number of the local port if the input parameter *rq* has the value TRUE. Within the parameter *chanCode* the communication channel code (*ETH1_uni0.*, ..., *ETH1_uni7*) is transferred by the function. Function returns the value TRUE if the requirement on new setup is accepted.

Parameter *remoteIP* determines the IP address of the station that the communication will be undertaken with. Parameter *remotePort* determines the port number where messages will be sent to. Parameter *localPort* determines which port will receive messages.

If the communication channel is set to communication via UDP protocol, the new IP address and ports will be set within the nearest PLC cycle turn. Afterwards, all UDP messages sent by the communication channel routed to the new IP address and new remote port. Between setup of the new target IP address and message sending to this address, at least one PLC cycle must be done. Also the message reception will run only from station which IP address corresponds to the newly set IP address and which uses corresponding port numbers.

If *remoteIP* address is 0.0.0.0, the channel receives a message from the station with any IP address providing that the port onto which the message is directed corresponds to the number entered in the parameter *localPort*. At the moment of the message reception the parameter *remoteIP* is set on the IP address of the station from which the message was received. Also, parameter *remotePort* changes the value according to the port number from which the message was sent. Therefore, after the reception of the UDP packet, the communication channel is set so that it is possible to answer to the message received.

If the communication channel is set onto the communication via the TCP protocol, new IP address and ports can be set providing that the connection is closed. Therefore, the settings can be changed if the TCP connection is setup. When the settings is changed, first it is necessary to terminate the connection. Further behaviour is similar as in the case of the UDP protocol.

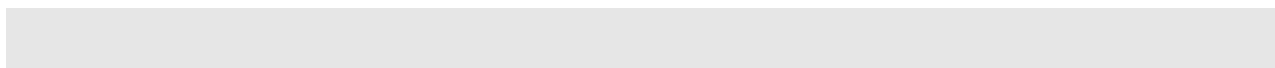
Message sending onto the address 0.0.0.0 is not allowed.

This function is supported on central units of the K rank (TC700 CP-7004, Foxtrot) from version v4.4. The function is supported for interface ETH1 only.

Variable description :

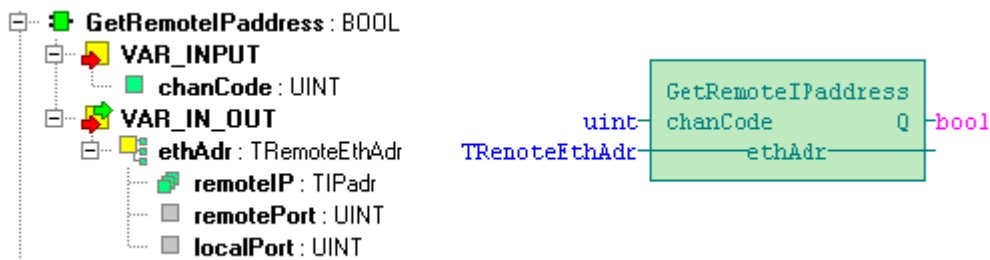
	<i>Variable</i>	<i>Type</i>	<i>Signification</i>
VAR_INPUT			
	<i>rq</i>	BOOL	Request on new IP address setup which the communication will run with. The entering edge on this input causes the initiation of the IP address change (if the new IP address is different from the current one)
	<i>chanCode</i>	USINT	Communication channel selection (ETH1_uni0, ..., ETH1_uni7)
VAR_IN_OUT			
	<i>EthAdr</i>	TRemoteEthAdr	IP address of the remote device and port numbers via which the communication is ran
	<i>.remoteIP</i>	TIPAdr	IP address of the device for communication
	<i>.remotePort</i>	UINT	Port number onto which messages will be sent, or rather, from which messages are received
	<i>.localPort</i>	UINT	Port number onto which messages will be sent, or rather, from which messages are received
SetRemoteIPAddress			
	<i>Release value</i>	BOOL	TRUE if the remote IP address is successfully set. Otherwise, FALSE

The example of the program with the *SetRemoteIPAddress* function call:



See also Function *GetRemoteIPAddress*

5.6 Function GetRemoteIPAddress

Library : *ComLib*

The *GetRemoteIPAddress* function returns currently set remote IP address, number of remote port and local port number.

This function is supported on central units of the K rank (TC700 CP-7004, Foxtrot) from version v4.4. The function is supported on the interface ETH1 only.

Variable description :

	Variable	Type	Signification
VAR_INPUT			
	<i>chanCode</i>	USINT	Communication channel selection (ETH1_uni0, ..., ETH1_uni7)
VAR_IN_OUT			
	<i>EthAdr</i>	TRemoteEthAdr	IP address of the remote device and port numbers via which the communication is ran
	<i>.remoteIP</i>	TIPadr	IP address of the device for communication
	<i>.remotePort</i>	UINT	Port number onto which messages will be sent, or rather, from which messages are received
	<i>.localPort</i>	UINT	Port number onto which messages will be sent, or rather, from which messages are received
GetRemoteIPAddress			
	<i>Release value</i>	BOOL	TRUE if the remote IP address is uploaded successfully. Otherwise, FALSE

The example of the program with the *GetRemoteIPAddress* function call:

See also Function SetRemoteIPAddress

[TECO_HTML_TO_HTML_GENERATOR]

[TITLE] 5.7 Funkce SetIPAddress [/TITLE]

[GROUP] Funkce [/GROUP]

[KEYWORDS] [/KEYWORDS]

[GLOBALS] [/GLOBALS]

[HIDDEN] SetIPAddress [/HIDDEN]

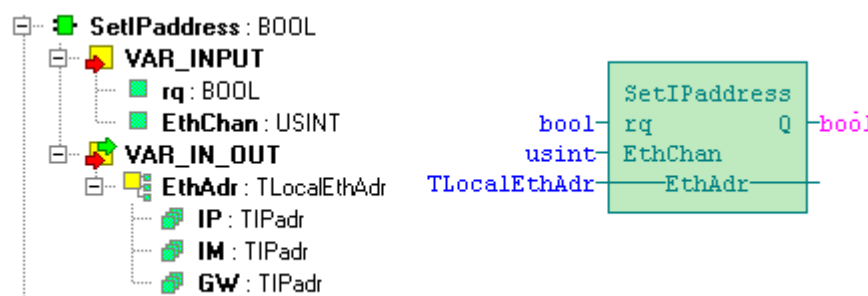
[HIDDEN_GLOBALS] [/HIDDEN_GLOBALS]

[NOTHING] [/NOTHING]

[/TECO_HTML_TO_HTML_GENERATOR]

5.7 Function SetIPAddress

Library : ComLib



The *SetIPAddress* function sets new local IP address, IP mask and gate address. New values are awaited in the variable *EthAdr* which is of the *TLocalEthAdr* type. The setup of the new address is done onto the entering edge of the input variable *rq*. The setup will take several PLC cycles and will return the TRUE value at the moment when the new IP address is set. All communication with the previous IP address will be cancelled. Simultaneously, the request on the automatic IP address obtaining from the DHCP server is cancelled (if it was on). Information on the actual status of the ethernet interface can be obtained anytime in the global variable *ETH1_STAT* or *ETH2_STAT*.

This function is supported on central units of the K rank (TC700 CP-7004, Foxtrot) from version v4.4.

Variable description :

	Variable	Type	Signification
VAR_INPUT			
	<i>rq</i>	BOOL	Request on new IP address setup The entering edge on this input causes the initialization of the IP address change (if the new IP address is different from the current one)
	<i>EthChan</i>	USINT	Ethernet channel specification (ETH1 for Ethernet on the central unit, ETH2 for Ethernet on the communication module)
VAR_IN_OUT			
	<i>EthAdr</i>	TLocalEthAdr	IP address, mask and gate

	<i>Variable</i>	<i>Type</i>	<i>Signification</i>
	<i>.IP</i>	TIPadr	IP address
	<i>.IM</i>	TIPadr	IP mask
	<i>.GW</i>	TIPadr	Gate address
SetIPAddress			
	<i>Release value</i>	BOOL	TRUE if the new IP address is set successfully. Otherwise, FALSE

The example of the program with the *SetIPAddress* function call:

```

PROGRAM prgTestSetIP
  VAR
    old_eth_adr   : TLocalEthAdr;
    new_eth_adr   : TLocalEthAdr;
    set, res, tmp : BOOL;
  END_VAR

  // read actual IP
  tmp := GetIPAddress( EthChan := ETH1, EthAdr := old_eth_adr);
  if new_eth_adr.IP[0] = 0 then
    new_eth_adr := old_eth_adr;           // use as init value for new adr
  end_if;

  // set new IP address if rq = TRUE
  res := SetIPAddress( rq := set, EthChan := ETH1, EthAdr := new_eth_adr);
  if res then set := FALSE; end_if;
END_PROGRAM

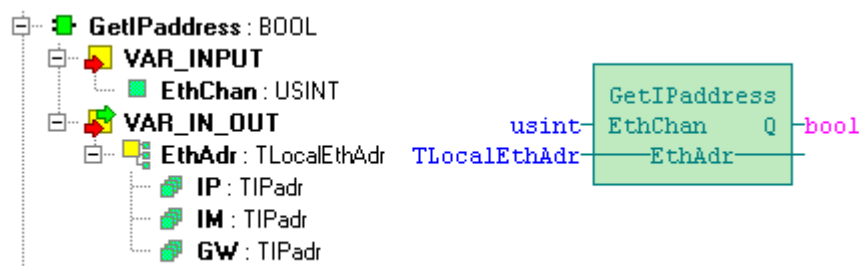
```

See also Function GetIPAddress, Type TLocalEthAdr, Function SetDHCPsupport

```
[TECO_HTML_TO_HTML_GENERATOR]
[TITLE] 5.8 Funkce GetIPAddress [/TITLE]
[GROUP] Funkce [/GROUP]
```

```
[KEYWORDS] [/KEYWORDS]
[GLOBALS] [/GLOBALS]
[HIDDEN] GetIPAddress [/HIDDEN]
[HIDDEN_GLOBALS] [/HIDDEN_GLOBALS]
[NOTHING] [/NOTHING]
[/TECO_HTML_TO_HTML_GENERATOR]
```

5.8 Function GetIPAddress

Library : *ComLib*

The *GetIPAddress* function returns the actual IP address, IP mask and gate address for the set ethernet channel. These values are saved into the variable *EthAdr* that has the structure of the *TLocalEthAdr* type. Declaration of this structure is a part of the library *ComLib*.

This function is supported on central units of the K rank (TC700 CP-7004, Foxtrot) from version v4.4.

Variable description :

	Variable	Type	Signification
VAR_INPUT			
	<i>EthChan</i>	USINT	Ethernet channel specification (ETH1 for Ethernet on the central unit, ETH2 for Ethernet on the communication module)
VAR_IN_OUT			
	<i>EthAdr</i>	TLocalEthAdr	Current IP address, mask and gate
	<i>.IP</i>	TIPadr	IP address
	<i>.IM</i>	TIPadr	IP mask
	<i>.GW</i>	TIPadr	Gate address
GetIPAddress			
	<i>Release value</i>	BOOL	TRUE if the IP address is uploaded successfully Otherwise, FALSE

The example of the program with the *GetIPAddress* function call:

```
PROGRAM prgTestGetIP
VAR
  old_eth_adr    : TLocalEthAdr;
  new_eth_adr    : TLocalEthAdr;
  set, res, tmp  : BOOL;
END_VAR

// read actual IP
tmp := GetIPAddress( EthChan := ETH1, EthAdr := old_eth_adr);
if new_eth_adr.IP[0] = 0 then
  new_eth_adr := old_eth_adr;           // use as init value for new adr
end_if;

// set new IP address if rq = TRUE
res := SetIPAddress( rq := set, EthChan := ETH1, EthAdr := new_eth_adr);
if res then set := FALSE; end_if;
END_PROGRAM
```

See also Function SetIPAddress, Type TLocalEthAdr

```

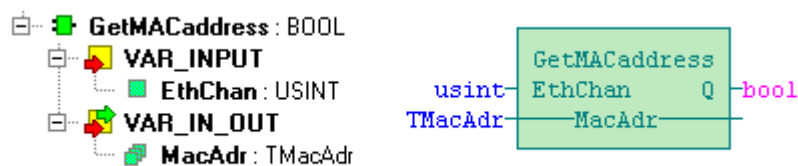
[TECO_HTML_TO_HTML_GENERATOR]
[TITLE] 5.9 Funkce GetMACaddress [/TITLE]
[GROUP] Funkce [/GROUP]

[KEYWORDS] [/KEYWORDS]
[GLOBALS] [/GLOBALS]
[HIDDEN] GetMACaddress [/HIDDEN]
[HIDDEN_GLOBALS] [/HIDDEN_GLOBALS]
[NOTHING] [/NOTHING]
[/TECO_HTML_TO_HTML_GENERATOR]

```

5.9 Function GetMACaddress

Library : *ComLib*



The *GetMACaddress* function returns the actual MAC address for the ethernet channel set. MAC address is saved in the variable *MacAdr* that has the structure of the *TmacAdr* type. Declaration of this structure is a part of the library *ComLib*. MAC address is a unique number in the whole world.

This function is supported on central units of the K rank (TC700 CP-7004, Foxtrot) from version v4.9.

Variable description :

	Variable	Type	Signification
VAR_INPUT			
	<i>EthChan</i>	USINT	Ethernet channel specification (ETH1 for Ethernet on the central unit, ETH2 for Ethernet on the communication module)
VAR_IN_OUT			
	<i>MacAdr</i>	TMacAdr	Current MAC address for the ethernet channel set
GetMACaddress			
	<i>Release value</i>	BOOL	TRUE the MAC address is uploaded successfully. Otherwise, FALSE

The example of the program with the *GetMACaddress* function call:

```

PROGRAM prgTestGetMAC
VAR
    mac_adr   : TMacAdr;
    tmp       : BOOL;
    message   : STRING;

```

```
END_VAR

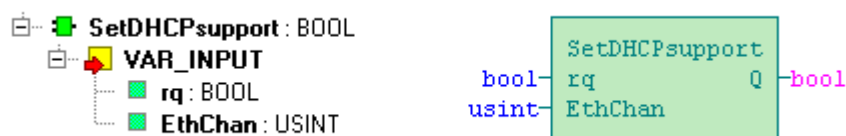
tmp := GetMACaddress( EthChan := ETH1, MacAdr := mac_adr);
IF (mac_adr[0] = 0) AND (mac_adr[1] = 16#0A) AND (mac_adr[2] = 16#14) THEN
    message := 'This is Teco device';
END_IF;
END_PROGRAM
```

See also Type TMacAdr

```
[TECO_HTML_TO_HTML_GENERATOR]
[TITLE] 5.10 Funkce SetDHCPsupport [/TITLE]
[GROUP] Funkce [/GROUP]
```

```
[KEYWORDS] [/KEYWORDS]
[GLOBALS] [/GLOBALS]
[HIDDEN] SetDHCPsupport [/HIDDEN]
[HIDDEN_GLOBALS] [/HIDDEN_GLOBALS]
[NOTHING] [/NOTHING]
[/TECO_HTML_TO_HTML_GENERATOR]
```

5.10 Function SetDHCPsupport

Library: *ComLib*

The *SetDHCPsupport* function switch on the function for automatic acquirement of the IP address of the PLC from the DHCP server within the PLC. Simultaneously, it terminates all currently running communications. The transfer to the automatic IP address will take several PLC cycles and function *SetDHCPsupport* returns the value TRUE at the moment when the transfer is ceased. Afterwards, PLC request the DHCP server on the IP address assignment that will be set on the ethernet interface set. From the moment when the transfer to the automatic IP address acquirement is initiated up to the moment of its assignment, the PLC operates only protocols ARP, ICMP and DHCP. If during the 4 seconds the IP address is not obtained (e.g. DHCP server is not present on the local network), the PLC use as an alternative configuration of IP address saved in the Ethernet interface configuration which is saved in the memory EEPROM. Information on actual status of the ethernet interface can be obtained anytime in the global variable *ETH1_STAT* or *ETH2_STAT*. The requirement on the automatic IP address acquirement is stored in the PLC even during the supply switch off. Therefore, after the supply is switched on, the PLC will again require the DHCP server for IP address assignment. The request on automatic IP address acquirement can be cancelled by the function *SetIPaddress* that will set firm IP address for the ethernet interface set.

DHCP server assigns IP address for a limited time only (typically one day). If the support of DHCP is switched on, PLC automatically requests for extension of the period which the IP address was assigned for. If this period can not be extended, PLC will keep the last assigned IP address and will set the global variable *ETH1_STAT.IP_expired* to the TRUE value.

This function is supported on central units of the K rank (TC700 CP-7004, Foxtrot) from version v4.9. The function is supported on the ETH1 interface only. Into the library ComLib is the function *SetDHCPsupport* located from ComLib_v13.

Variable description :

	Variable	Type	Signification
VAR_INPUT			
	<i>rq</i>	BOOL	Reques on switch on of support of the DHCP protocol The entering endge on this input invokes the termination of all communications with the current IP address and initiates the

	Variable	Type	Signification
			request on assignment of a new IP address on the DHCP server
	<i>EthChan</i>	USINT	Specification of the ethernet channel (ETH1 for Ethernet on the central unit)
SetDHCPsupport			
	<i>Release value</i>	BOOL	TRUE, when the transfer to automatic acquirement of IP address of the PLC from the DHCP server is successful, otherwise, FALSE Value TRUE means that the PLC will try to obtain the IP address from the DHCP server. Information whether the address was obtained is saved in the global variable <i>ETH1_STAT.IP_obtained</i>

The example of the program with the *SetDHCPsupport* function call:

```

PROGRAM prgTestDHCPsupport
VAR
  set_fix_IP : BOOL;           // request for fixed IP address
  set_aut_IP : BOOL;           // request for obtaining IP address from DHCP
  my_IP      : TLocalEthAdr;
  info       : STRING;
  fix_IP     : TLocalEthAdr := (IP := [192,168,1,1],
                                IM := [255,255,255,0],
                                GW := [192,168,1,200]);

END_VAR

IF set_fix_IP THEN
  IF SetIPAddress(rq:= set_fix_IP, EthChan:= ETH1, EthAdr:= fix_IP) THEN
    set_fix_IP := FALSE;
  END_IF;
ELSE
  IF set_aut_IP THEN
    IF SetDHCPsupport(rq := set_aut_IP, EthChan := ETH1) THEN
      set_aut_IP := FALSE;
    END_IF;
  END_IF;
END_IF;

GetIPAddress(EthChan := ETH1, EthAdr := my_IP);
IF ETH1_STAT.DHCP_enabled AND ETH1_STAT.IP_obtained THEN
  info := 'DYNAMIC IP:' + IPADR_TO_STRING(IPAdr := my_IP.IP);
ELSE
  info := 'FIXED IP:' + IPADR_TO_STRING(IPAdr := my_IP.IP);
END_IF;
END_PROGRAM

```

The program sets the firm IP address 192.168.1.1 on the ETH1 interface in case that the TRUE is entered into the variable *set_fix_IP* (e.g. From the WebMakeru). If we enter TRUE into the variable *set_aut_IP*, then the program will try to obtain the IP address from DHCP server. The actually set IP address can be checked out in the variable *info*.

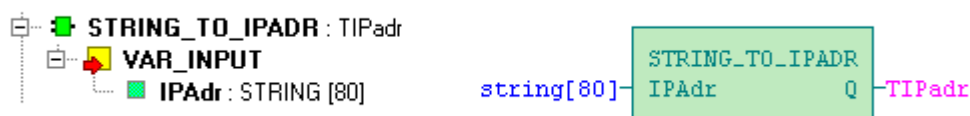
See also Function SetIPAddress, Global variables

```
[TECO_HTML_TO_HTML_GENERATOR]
[TITLE] 5.11 Funkce STRING_TO_IPADR [/TITLE]
[GROUP] Funkce [/GROUP]
```

```
[KEYWORDS] [/KEYWORDS]
[GLOBALS] [/GLOBALS]
[HIDDEN] STRING_TO_IPADR [/HIDDEN]
[HIDDEN_GLOBALS] [/HIDDEN_GLOBALS]
[NOTHING] [/NOTHING]
[/TECO_HTML_TO_HTML_GENERATOR]
```

5.11 Function *STRING_TO_IPADR*

Library : *ComLib*



Function *STRING_TO_IPADR* undertakes the conversion of the IP address registered in the variable of the STRING type into the structure of the TIPadr type. Function awaits IP address in the common form e.g. '192.168.33.1', so as decimal numbers separated by a dot. Head zero are admissible e.g. '192.168.033.001'. Spaces within the string are not supported.

The function *STRING_TO_IPADR* is located into the library ComLib from version ComLib_v13.

Variable description :

	<i>Variable</i>	<i>Type</i>	<i>Signification</i>
VAR_INPUT			
	<i>IPAdr</i>	STRING	IP address, string of characters containing 4 decimal numbers separated by dots
STRING_TO_IPADR			
	<i>Release value</i>	TIPadr	Input string transferred onto the array with 4 elements USINT If the transfer is not successful, the function returns ANY_IP i.e. 0.0.0.0

The example of the program with the *STRING_TO_IPADR* function call:

```
PROGRAM prgTest_STRING_TO_IPADR
VAR
  tst_IP      : TLocalEthAdr;
  IP_addr     : STRING[16] := '192.168.1.1';
  IP_mask     : STRING[16] := '192.168.1.13';
  GW_addr     : STRING[16] := '192.168.1.200';
END_VAR

tst_IP.IP := STRING_TO_IPADR(IPAdr := IP_addr);
```

```
tst_IP.IM := STRING_TO_IPADR(IPAdr := IP_mask);  
tst_IP.GW := STRING_TO_IPADR(IPAdr := GW_addr);  
END_PROGRAM
```

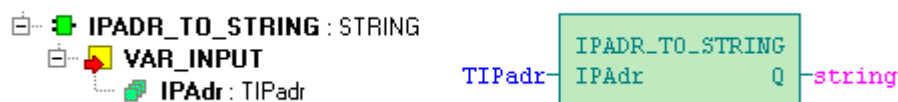
See also Function IPADR_TO_STRING

```
[TECO_HTML_TO_HTML_GENERATOR]
[TITLE] 5.12 Funkce IPADR_TO_STRING [/TITLE]
[GROUP] Funkce [/GROUP]
```

```
[KEYWORDS] [/KEYWORDS]
[GLOBALS] [/GLOBALS]
[HIDDEN] IPADR_TO_STRING [/HIDDEN]
[HIDDEN_GLOBALS] [/HIDDEN_GLOBALS]
[NOTHING] [/NOTHING]
[/TECO_HTML_TO_HTML_GENERATOR]
```

5.12 Function IPADR_TO_STRING

Library : *ComLib*



The *IPADR_TO_STRING* function undertakes the conversion of the IP address registered in the variable of the *TIPadr* type into the variable of the *STRING* type.

The function *IPADR_TO_STRING* is located into the library *ComLib* from version *ComLib_v13*.

Description variable :

	Variable	Type	Signification
VAR_INPUT			
	<i>IPAdr</i>	<i>TIPadr</i>	IP address entered as an array with 4 elements of the <i>USINT</i> type
IPADR_TO_STRING			
	<i>Release value</i>	<i>STRING</i>	The string of characters containing 4 decimal numbers separated by dots

The example of the program with the *IPADR_TO_STRING* function call:

```

PROGRAM prgTest_IPADR_TO_STRING
VAR
  my_IP      : TLocalEthAdr;
  info       : STRING;
END_VAR

GetIPAddress(EthChan := ETH1, EthAdr := my_IP);
info := ' IP: ' + IPADR_TO_STRING(IPAdr := my_IP.IP) +
        ' IM: ' + IPADR_TO_STRING(IPAdr := my_IP.IM) +
        ' GW: ' + IPADR_TO_STRING(IPAdr := my_IP.GW);
END_PROGRAM
  
```

See also Function *STRING_TO_IPADR*

[TECO_HTML_TO_HTML_GENERATOR]

[TITLE] 6 Úvod [/TITLE]

[GROUP] Funkční bloky [/GROUP]

[KEYWORDS] [/KEYWORDS]

[GLOBALS] [/GLOBALS]

[HIDDEN] [/HIDDEN]

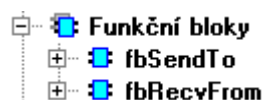
[HIDDEN_GLOBALS] [/HIDDEN_GLOBALS]

[NOTHING] [/NOTHING]

[/TECO_HTML_TO_HTML_GENERATOR]

6 **FUNCTION BLOCKS**

The ComLib library contains following function blocks:



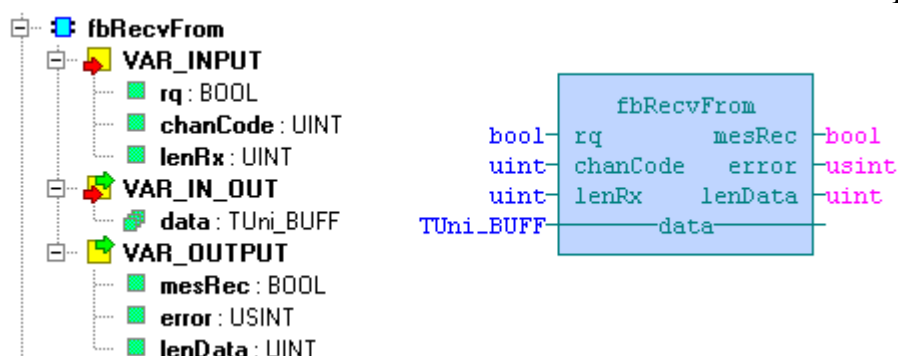
- *fbRecvFrom* Function block for message receiving pro příjem zpráv
- *fbSendTo* Function block for message sending

[TECO_HTML_TO_HTML_GENERATOR]
 [TITLE] 6.1 Funkční blok fbRecvFrom [/TITLE]
 [GROUP] Funkční bloky [/GROUP]

[KEYWORDS] [/KEYWORDS]
 [GLOBALS] [/GLOBALS]
 [HIDDEN] fbRecvFrom [/HIDDEN]
 [HIDDEN_GLOBALS] [/HIDDEN_GLOBALS]
 [NOTHING] [/NOTHING]
 [/TECO_HTML_TO_HTML_GENERATOR]

6.1 Function block fbRecvFrom

Library : ComLib



The *fbRecvFrom* function block is used for message reception from the serial channel or from the ethernet. The communication channel must be set to the *uni* mode.

This function block is supported on central units of the K rank (TC700 CP-7004, Foxtrot) from version v4.4.

Variable description :

	Variable	Type	Signification
VAR_INPUT			
	<i>rq</i>	BOOL	Control variable. If it is TRUE, the function block receives data from the communication channel, if it is FALSE the data reception is forbidden
	<i>chanCode</i>	UINT	Communication channel code <i>ETH1_uni0</i> ethernet channel ETH1, connection uni0 <i>ETH1_uni7</i> ethernet channel ETH1, connection uni7 <i>ETH2_uni0</i> ethernet channel ETH2, connection uni0 <i>ETH2_uni7</i> ethernet channel ETH2, connection uni7 <i>CH1_uni</i> sériový channel CH1, mode uni <i>CH10_uni</i> sériový channel CH10, mode uni
	<i>lenRx</i>	UINT	Max. length of data received (size of variable which the received message will be saved into)
VAR_IN_OUT			

	<i>Variable</i>	<i>Type</i>	<i>Signification</i>
	<i>data</i>	TUniBuf	Variable where the received message will be saved into
VAR_OUTPUT			
	<i>mesRec</i>	BOOL	The flag of the received message If it is TRUE, new message was received.
	<i>error</i>	USINT	Error code. If it is 0 (COM_OK), the reception passed without an error In case of an error, the codes COM_ERR1, ..., COM_ERR64 are returned
	<i>lenData</i>	UINT	Length of the received message (number of bytes)

In the following example there is a function block *fbRecvFrom* used for the message reception from the serial channel CH1. The reception of text messages with the max size of 100 characters is expected. The setup of transmission parameters of the serial channel is apparent from the following dialogue.

Nastavení univerzálního režimu kanálu

Přijímací zóna Délka zóny: 101 Adresa zóny: ☐ 4 Přijímací zóna: CH2_ZoneIN	Vysílací zóna Délka zóny: 101 Adresa zóny: ☐ 4 Vysílací zóna: CH2_ZoneOUT	Komunikační rychlost: 38 400 Formát dat: 8b, bez parity
Počáteční znak ☐ Detekovat ☐ Vysílat Kód znaku: 0	Koncový znak ☐ Detekovat ☐ Vysílat ☐ Dva znaky Kód znaku: 0 0	Adresa stanice Adresa stanice: 0 ☐ Detekovat při příjmu ☐ Zápis při vysílání
Parita prvního bytu přijímané zprávy ☒ Stejná parita jako u ostatních ☐ Opačná parita než u ostatních	Parita prvního bytu vysílané zprávy ☒ Stejná parita jako u ostatních ☐ Opačná parita než u ostatních	Kontrolní součet ☐ Kontrola při příjmu ☐ Výpočet při vysílání Poz. prvního znaku CHS: 0
Potvrzení zprávy bez dat ☐ Detekovat ☐ Vysílat ☐ Dva znaky Kód znaku: 0 0	Délka zprávy ☐ Detekovat při příjmu ☐ Zápis při vysílání Pozice délky zprávy: 0 Maximální délka: 0	Režim řízení modemových signálů Řízení signálu RTS: automatická hodnota Řízení signálu DTR: trvale hodnota 0 ☐ Odpojení přijímače během vysílání
Min. doba klidu na lince mezi přijímanými zprávami (počet bytů): 10 Min. doba klidu na lince mezi vysílanými zprávami (počet bytů): 10		OK Zrušit Nápověda

```

PROGRAM ExampleRecvFrom
VAR
  RecvFromCH1 : fbRecvFrom;
  rxBuf       : STRING [100];
  errMsg      : STRING;
END_VAR

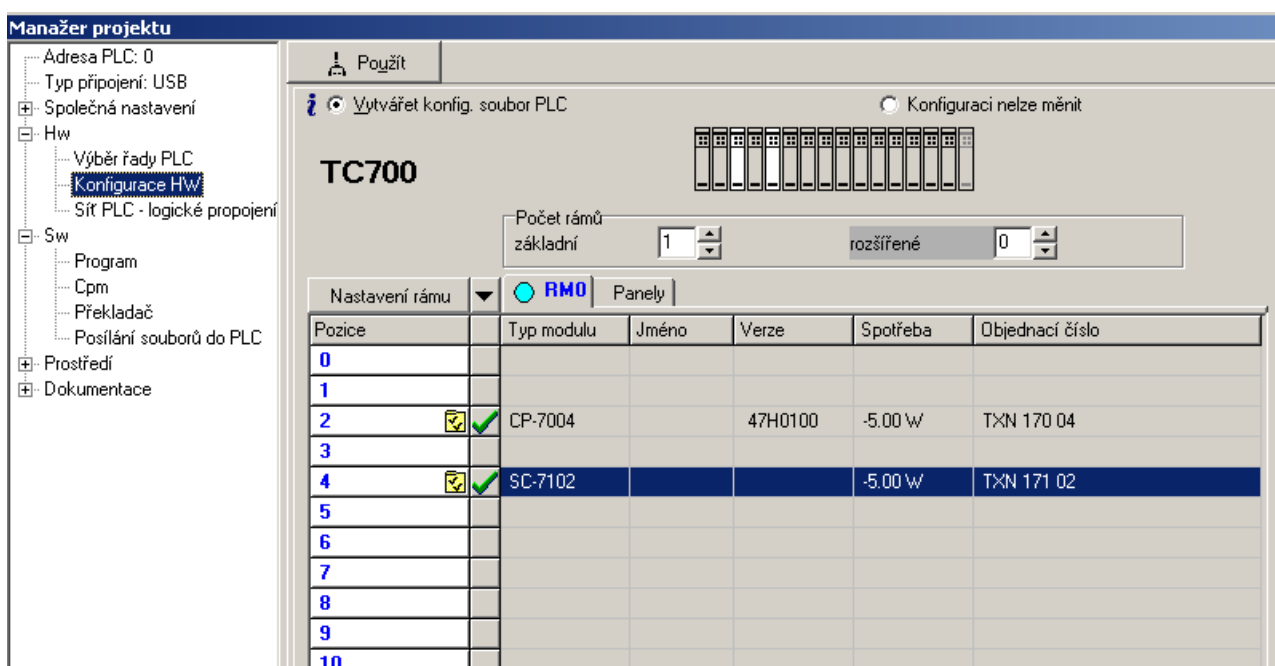
```

```

// receiving
RecvFromCH1( rq := TRUE, chanCode := CH1_uni,
             lenRx := 100, data := void(rxBuf));
if RecvFromCH1.mesRec then
  // new message received
  if RecvFromCH1.error = 0 then
    // process new message
    // ...
  else
    errMsg := GetLastComErrTxt( RecvFromCH1.error); // show error as a text
  end_if;
end_if;
END_PROGRAM

```

The following example shows the use of function block *fbRecvFrom* for reception of messages via TCP protocol. First of all, it is necessary to set the uni mode for ethernet interface. The setup is undertaken in the Mosaic environment in the Project manager in the node of HW configuration. In this concrete case, the interface ETH2 was used on the module SC-7102.



By clicking on the yellow icon of the module SC-7102 (see the highlighted line in the following picture) the setting of communication for module SC-7102 is initiated. In this dialogue, it is necessary first to switch on the uni mode of the channel on the ethernet interface so, that we click on the line ETH uni-off (highlighted in the following picture) and then we switch the channel mode in the left upper corner from the value OFF to uni.

Nastavení parametrů kanálů

Nastavení kom. kanálů se nese s programem a je nadřazeno nastavení v EEPROM CPM !

Režim kanálu: **OFF**
Číslování kanálů: **OFF**
Adresa pro komunikaci: 0
Komunikační rychlost:
Prodleva odpovědi: 0
Dopravní zpoždění: 0
Detekce CTS:
Předávání tokenů:
Přenos s paritou:
Ethernet:
Adresa IP: 192.168.033.169
Maska podsítě: 255.255.255.000
Výchozí brána: 192.168.134.203

Načíst z PLC
Uložit do PLC

Zálohovat program do EEPROM: off

OK Zrušit Nápověda

Struktura kanálů	rám / pozice	Režim kanálu	Adresa pro komunikaci	Komunikační rychlost	Prodleva odpovědi	Dopravní zpoždění	Detekce CTS	Předávání tokenů	Přenos s paritou
CP-7004	0 / 2								
SC-7102	0 / 4								
CH									
CH3		OFF							
CH4		OFF							
ETH2			192.168.033.170						
ETH		PC							
ETH		PLC -off							
ETH		uni -off							

After setting to uni mode, the dialogue should appear as follows

Nastavení parametrů kanálů

Nastavení kom. kanálů se nese s programem a je nadřazeno nastavení v EEPROM CPM !

Režim kanálu: **uni**
Číslování kanálů:
Adresa pro komunikaci: 0
Komunikační rychlost:
Prodleva odpovědi: 0
Dopravní zpoždění: 0
Detekce CTS:
Předávání tokenů:
Přenos s paritou:
Ethernet:
Adresa IP: 192.168.033.169
Maska podsítě: 255.255.255.000
Výchozí brána: 192.168.134.203

Načíst z PLC
Uložit do PLC

Zálohovat program do EEPROM: off

OK Zrušit Nápověda

Struktura kanálů	rám / pozice	Režim kanálu	Adresa pro komunikaci	Komunikační rychlost	Prodleva odpovědi	Dopravní zpoždění	Detekce CTS	Předávání tokenů	Přenos s paritou
CP-7004	0 / 2								
SC-7102	0 / 4								
CH									
CH3		OFF							
CH4		OFF							
ETH2			192.168.033.170						
ETH		PC							
ETH		PLC -off							
ETH		uni							

By clicking on the yellow icon on the line ETH uni (highlighted in the previous picture) the next dialogue is initiated where maximum size of sending and receiving zone is set (number of bytes), the type of the TCP slave protocol is selected (PLC will not setup the TCP connection actively) and the port number is set where data will be awaited (local port).

The following program receives data broadcasted by the TCP protocol on the port 61001. Received data are saved in the variable `rxData`. Establishment and termination of the TCP connection is controlled by the station that is sending data. This station can be for example PLC with the program `prgTestSendTCP` which is stated in examples in the following chapter.

```

PROGRAM prgTestRecvTCP
VAR
    recvTim      : TON;
    recvFrom     : fbRecvFrom;
    rxData       : ARRAY[0..49] OF USINT;
    cntOK        : UDINT;
    cntERR       : UDINT;
    lastErr      : USINT;
    lastErrTxt   : STRING;
END_VAR

recvTim(IN := TRUE, PT := T#35s);
IF recvTim.Q THEN
    cntERR := cntERR + 1; recvTim( IN := false);
END_IF;
recvFrom( rq := true, chanCode := ETH2_uni0,
          lenRx := sizeof(rxData), data := void(rxData),
          error => lastErr);

IF recvFrom.mesRec THEN
    recvTim( IN := false);
    IF recvFrom.error = COM_OK THEN
        cntOK := cntOK + 1;
        // process incoming data (see rxData[])
        // ...
    ELSE
        cntERR := cntERR + 1;
        lastErrTxt := GetLastComErrTxt(errCode := lastErr);
    END_IF;
END_IF;
END_PROGRAM

```

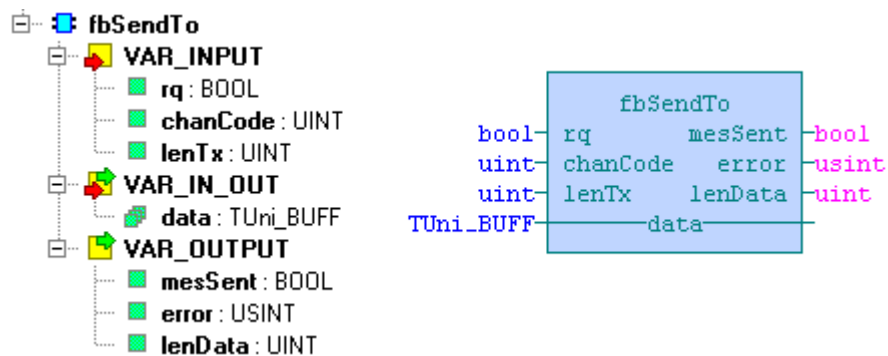
See also Function block fbSendTo

[TECO_HTML_TO_HTML_GENERATOR]
 [TITLE] 6.2 Funkční blok fbSendTo [/TITLE]
 [GROUP] Funkční bloky [/GROUP]

[KEYWORDS] [/KEYWORDS]
 [GLOBALS] [/GLOBALS]
 [HIDDEN] fbSendTo [/HIDDEN]
 [HIDDEN_GLOBALS] [/HIDDEN_GLOBALS]
 [NOTHING] [/NOTHING]
 [/TECO_HTML_TO_HTML_GENERATOR]

6.2 Function block fbSendTo

Library : ComLib



The *fbSendTo* function block is used for message sending form the serial channel or ethernet. Communication channel must be set to *uni* mode.

This function block is supported on central units of the K rank (TC700 CP-7004, Foxtrot) from version v4.4.

Variable description :

	Variable	Type	Signification
VAR_INPUT			
	<i>rq</i>	BOOL	Control variable. If it is TRUE, the function block broadcasts data into the communication channel
	<i>chanCode</i>	UINT	Communication channel code <i>ETH1_uni0</i> ethernet channel ETH1, connection uni0 <i>ETH1_uni7</i> ethernet channel ETH1, connection uni7 <i>ETH2_uni0</i> ethernet channel ETH2, connection uni0 <i>ETH2_uni7</i> ethernet channel ETH2, connection uni7 <i>CH1_uni</i> serial channel CH1, mode uni <i>CH10_uni</i> serial channel CH10, mode uni
	<i>lenTx</i>	UINT	Sent message lenght (number of bytes)
VAR_IN_OUT			

	<i>Variable</i>	<i>Type</i>	<i>Signification</i>
	<i>data</i>	TUniBuf	Variable where the broadcasted message is ready
VAR_OUTPUT			
	<i>mesSent</i>	BOOL	Sent message flag If it is TRUE, the sending was commenced
	<i>error</i>	USINT	Error code. If it is 0 (COM_OK), all ran errorless In case of an error, it returns codes COM_ERR1, ..., COM_ERR64
	<i>lenData</i>	UINT	Sent message lenght (number of bytes)

In the following example the function block *fbSendTo* is used for cyclic broadcasting of the message by the seriál channel CH1. The text „Message number : 1“ is sent as a message. The number stated in the string is incremented during each message sending so, that it indicates the number of messages sent. The rythm of sending is controlled by the timer *sendTim*. The particular setup of the seriál channel is shown in the following picture.

Nastavení univerzálního režimu kanálu

Přijímací zóna Délka zóny: 101 Adresa zóny: 4 Přijímací zóna: CH2_ZoneIN	Vysílací zóna Délka zóny: 101 Adresa zóny: 4 Vysílací zóna: CH2_ZoneOUT	Komunikační rychlost: 38 400 Formát dat: 8b, bez parity
Počáteční znak ☐ Detekovat ☐ Vysílat Kód znaku: 0	Koncový znak ☐ Detekovat ☐ Vysílat ☐ Dva znaky Kód znaku: 0 0	Adresa stanice Adresa stanice: 0 ☐ Detekovat při příjmu ☐ Zápis při vysílání
Parita prvního bytu přijímané zprávy ☒ Stejná parita jako u ostatních ☐ Opačná parita než u ostatních	Parita prvního bytu vysílané zprávy ☒ Stejná parita jako u ostatních ☐ Opačná parita než u ostatních	Kontrolní součet ☐ Kontrola při příjmu ☐ Výpočet při vysílání Poz. prvního znaku CHS: 0
Potvrzení zprávy bez dat ☐ Detekovat ☐ Vysílat ☐ Dva znaky Kód znaku: 0 0	Délka zprávy ☐ Detekovat při příjmu ☐ Zápis při vysílání Pozice délky zprávy: 0 Maximální délka: 0	Režim řízení modemových signálů Řízení signálu RTS: automatická hodnota Řízení signálu DTR: trvale hodnota 0 ☐ Odpojení přijímače během vysílání
Min. doba klidu na lince mezi přijímanými zprávami (počet bytů): 10 Min. doba klidu na lince mezi vysílanými zprávami (počet bytů): 10		OK Zrušit Nápověda

```

PROGRAM ExampleSendTo
VAR
  SendToCH1      : fbSendTo;
  txBuf          : STRING[100];
  sendTim        : TON;
  sendCnt        : UDINT;
  errMsg         : STRING;
END_VAR

sendTim(IN := TRUE, PT := T#3s);

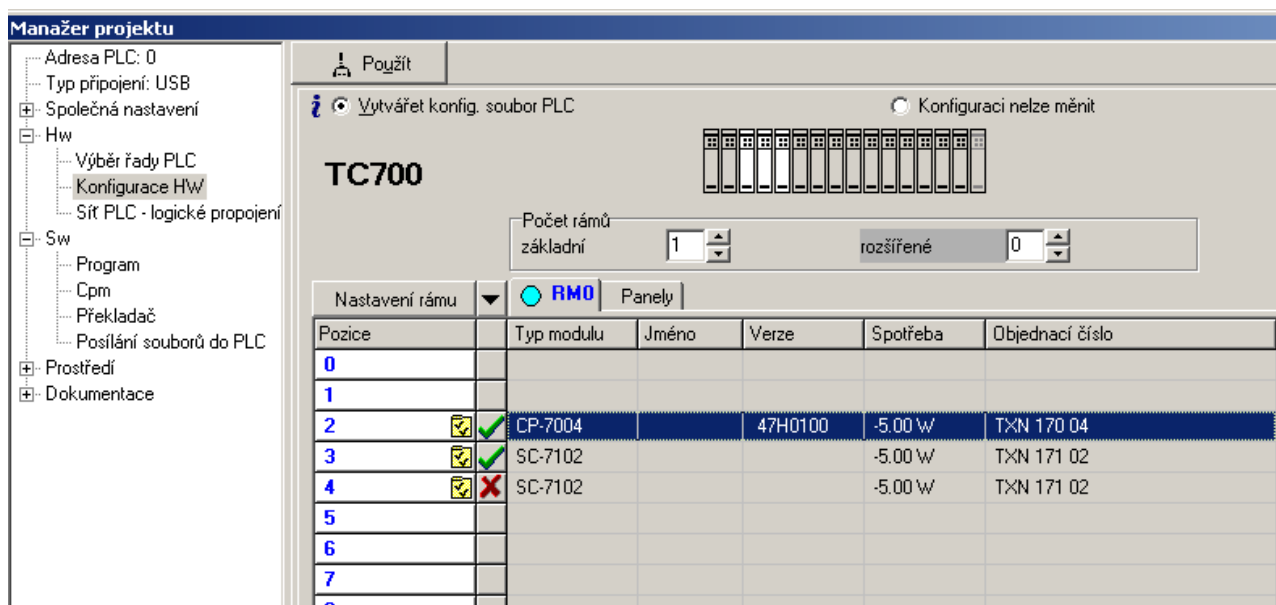
```

```

if sendTim.Q then                                     // send new message every 3 sec
    sendCnt := sendCnt + 1;                           // number of messages
    txBuf := 'Message number : ' + UDINT_TO_STRING( sendCnt);
    SendToCH1( rq := TRUE, chanCode := CH1_uni,
               lenTx := len(txBuf), data := void(txBuf));
    if SendToCH1.error = 0 then                       // no error
        if SendToCH1.mesSent then                   // message sent succesfully
            sendTim(IN := FALSE);                   // timer restart
        end_if;
    else
        errMsg := GetLastComErrTxt( SendToCH1.error); // show error as a text
    end_if;
end_if;
END_PROGRAM

```

Another example shows the use of the function block *fbSendTo* for data sending by TCP protocol via the interface ETH1. Program establishes the connection via TCP protocol every 30 seconds, sends data and closes the connection again. The target IP address and port numbers are, in this case, firmly set by the ethernet channel settings. The setup is undertaken in the Mosaic environment in the Project manager in the node of HW configuration.



By clicking on the yellow icon on the line with the central unit (here CP-7004), the dialogue for ethernet interface setup is opened. In this dialogue, it is necessary first to switch on the uni mode of the channel on the ethernet interface so, that we click on the line ETH uni-off (highlighted in the following picture) and then we switch the channel mode in the left upper corner from the value OFF to uni.

Nastavení parametrů kanálů

Nastavení kom. kanálů se nese s programem a je nadřazeno nastavení v EEPROM CPM !

Režim kanálu: OFF
Číslování kanálů: OFF
Adresa pro komunikaci: 0
Komunikační rychlost:
Prodleva odpovědi: 0
Dopravní zpoždění: 0
Detekce CTS:
Předávání tokenů:
Přenos s paritou:
Ethernet:
Adresa IP: 192.168.033.169
Maska podsítě: 255.255.255.000
Výchozí brána: 192.168.134.203

Načíst z PLC
Uložit do PLC

Zálohovat program do EEPROM: off

OK Zrušit nápověda

Struktura kanálů	rám / pozice	Režim kanálu	Adresa pro komunikaci	Komunikační rychlost	Prodleva odpovědi	Dopravní zpoždění	Detekce CTS	Předávání tokenů	Přenos s paritou
CP-7004	0 / 2								
CH									
CH1		uni							
CH2		uni							
ETH1			192.168.033.169						
ETH		PC, MDB							
ETH		PLC -off							
ETH		uni -off							
ETH		BAC -off							
USB									
USB		PC	0						
SC-7102	0 / 3								
SC-7102	0 / 4								

After setting to uni mode, the dialogue should appear as follows

Nastavení parametrů kanálů

Nastavení kom. kanálů se nese s programem a je nadřazeno nastavení v EEPROM CPM !

Režim kanálu: uni
Číslování kanálů:
Adresa pro komunikaci: 0
Komunikační rychlost:
Prodleva odpovědi: 0
Dopravní zpoždění: 0
Detekce CTS:
Předávání tokenů:
Přenos s paritou:
Ethernet:
Adresa IP: 192.168.033.169
Maska podsítě: 255.255.255.000
Výchozí brána: 192.168.134.203

Načíst z PLC
Uložit do PLC

Zálohovat program do EEPROM: off

OK Zrušit nápověda

Struktura kanálů	rám / pozice	Režim kanálu	Adresa pro komunikaci	Komunikační rychlost	Prodleva odpovědi	Dopravní zpoždění	Detekce CTS	Předávání tokenů	Přenos s paritou
CP-7004	0 / 2								
CH									
CH1		uni							
CH2		uni							
ETH1			192.168.033.169						
ETH		PC, MDB							
ETH		PLC -off							
ETH		uni							
ETH		BAC -off							
USB									
USB		PC	0						
SC-7102	0 / 3								
SC-7102	0 / 4								

By clicking on the yellow icon on the line ETH uni (highlighted in the previous picture) the next dialogue is initiated where maximum size of sending and receiving zone is set (number of bytes), the type of the TCP master protocol is selected (PLC will setup the TCP connection actively) and the IP address of the system is set where data will be sent to. The remote port is a port where data will be sent to, the local port is, on the contrary, a port where data are sent from.

Program that sends data cyclically every 30 seconds onto the address 192.168.33.171 and remote port 61001 appears as follows.

```

TYPE
  states : (idle, estabCon, sendData, closeCon);
END_TYPE

PROGRAM prgTestSendTCP
VAR
  sendTim      : TON;
  chkTim       : TON;
  state        : states := idle;
  txData       : ARRAY[0..49] OF USINT;
  sendTo       : fbSendTo;
  cntOK        : UDINT;
  cntERR       : UDINT;
  lastErr      : USINT;
  lastErrTxt   : STRING;
END_VAR

sendTim(IN := true, PT := T#30s);
CASE state OF
  idle :
    IF IsEstabTCPconnection(chanCode := ETH1_UNIO) THEN
      CloseTCPconnection(chanCode := ETH1_UNIO);
    END_IF;
    IF sendTim.Q THEN
      state := estabCon; sendTim(IN := false); chkTim(IN := false);
    END_IF;

  estabCon :
    lastErr := EstabTCPconnection(chanCode := ETH1_UNIO);
    IF lastErr = COM_OK THEN
      state := sendData;
    END_IF;

  sendData :
    IF IsEstabTCPconnection(chanCode := ETH1_UNIO) THEN

```

```
// prepare outgoing data (see txData)
txData[0] := txData[0] + 1;
sendTo( rq := true, chanCode := ETH1_UNI0,
        lenTx := sizeof(txData), data := void(txData),
        error => lastErr);
IF sendTo.mesSent THEN
    cntOK := cntOK + 1;
END_IF;
state := closeCon;
ELSE
    chkTim(IN := true, PT := T#20s);
    IF chkTim.Q THEN
        lastErr := COM_ERR64;
        state := closeCon; cntERR := cntERR + 1;
    END_IF;
END_IF;

closeCon :
    CloseTCPconnection(chanCode := ETH1_UNI0);
    state := idle;
END_CASE;

lastErrTxt := GetLastComErrTxt(errCode := lastErr);
END_PROGRAM
```

See also Function block fbRecvFrom
[TECO_HTML_TO_HTML_GENERATOR]
[/TECO_HTML_TO_HTML_GENERATOR]