MQTTLib Library

TXV 005 38.02
Fourth edition
February 2021

Modifications history

Date	Publication	Modification description
February 2018	1	First version
April 2018	2	Application profile is not needed
May 2020	3	Added error descriptions
February 2021	4	Added implementation of LWT functions, library version MQTTLib v1.5

Content

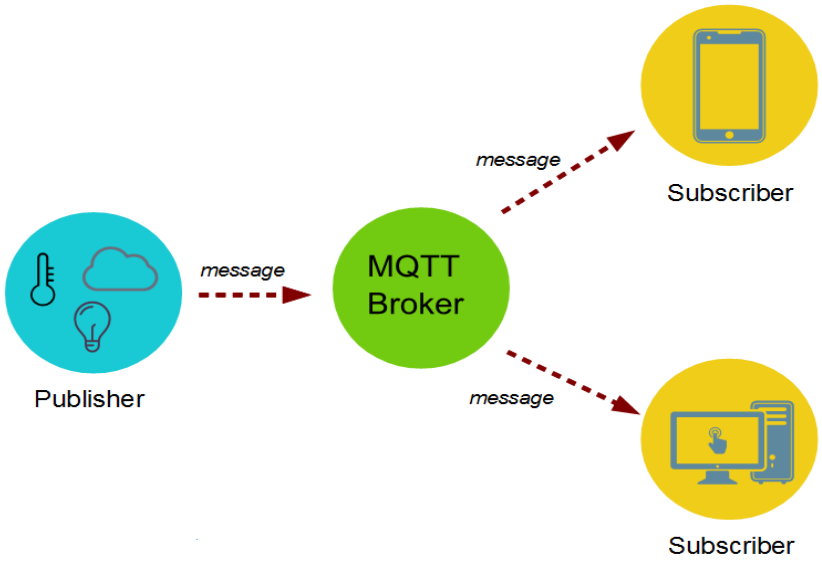
1 INTRODUCTION

Library: MQTTLib

MQTT is a Client Server publish/subscribe messaging transport protocol. It is light weight, open, simple, and designed so as to be easy to implement. These characteristics make it ideal for use in many situations, including constrained environments such as for communication in Machine to Machine (M2M) and Internet of Things (IoT) contexts where a small code footprint is required and/or network bandwidth is at a premium.

The protocol runs over TCP/IP, or over other network protocols that provide ordered, lossless, bi-directional connections. Its features include:

- Use of the publish/subscribe message pattern which provides one-to-many message distribution and decoupling of applications.
- A messaging transport that is agnostic to the content of the payload.
- Three qualities of service for message delivery:
  - "**At most once**", (QoS 0) where messages are delivered according to the best efforts of the operating environment. Message loss can occur. This level could be used, for example, with ambient sensor data where it does not matter if an individual reading is lost as the next one will be published soon after.
  - "**At least once**", (QoS 1) where messages are assured to arrive but duplicates can occur.
  - "**Exactly once**", (QoS 2) where message are assured to arrive exactly once. This level could be used, for example, with billing systems where duplicate or lost messages could lead to incorrect charges being applied.
- A small transport overhead and protocol exchanges minimized to reduce network traffic.
- A mechanism to notify interested parties when an abnormal disconnection occurs.



If required to use MQTTLib library functions in the PLC application program, you must add this library to your project. Along with the library in the project MQTTLib automatically added following libraries ComLib, StdLib, InternetLib a SysLib because MQTTLib uses some functions from these libraries. The library is supplied as part of the installation of the Mosaic version v2017.1.

MQTTLib library is not supported on systems with TC-650 for the system TC700 library is not compatible with the processor modules CP-7002, CP-7003 and CP-7005.
MQTTLib library functions are supported in the central units of series K and L (TC700 CP-7000, CP-7004 and CP-7007, all variants of Foxtrot) from version v10.1.

Documentation number for library MQTTLib is TXV 005 38.

1.1 Message Brokering Basic Terms

Library: MQTTLib

- **Broker**: The broker accepts messages from clients and then delivers them to any interested clients. Messages belong to a topic. (Sometimes brokers are called “servers.”)
- **Client**: A “device” that either publishes a message to a topic, subscribes to a topic, or both.
- Topic: A namespace (or place) for messages on the broker. Clients subscribe and publish to a topic.
- **Publish**: A client sending a message to the broker, using a topic name.
- **Subscribe**: A client tells the broker which topics interest it. Once subscribed, the broker sends messages published to that topic. (In some configurations the broker sends “missed” messages.) A client can subscribe to multiple topics.
- **Unsubscribe**: Tell the broker you are bored with this topic. In other words, the broker will stop sending messages on this topic.

1.2 Topic Basic Terms

Library: MQTTLib

A topic is a simple string that can have more hierarchy levels, which are separated by a slash. A sample topic for sending temperature data of the living room could be house/living-room/temperature. On one hand the client can subscribe to the exact topic or on the other hand use a wildcard. The subscription to house/+/temperature would result in all message send to the previously mention topic house/living-room/temperature as well as any topic with an arbitrary value in the place of living room, for example house/kitchen/temperature. The plus sign is a single level wild card and only allows arbitrary values for one hierarchy.

If you need to subscribe to more than one level, for example to the entire subtree, there is also a multilevel wildcard (#). It allows to subscribe to all underlying hierarchy levels. For example house/# is subscribing to all topics beginning with house.

1.3 Development tools

Library: MQTTLib

In this protocol, the central communication point is the MQTT broker. It is in charge of managing all messages between the senders and the receivers. To interact with an MQTT broker, you'll need an MQTT client, which is the one in charge of publishing/subscribing messages to the broker. The MQTT client includes a topic into the message. It is in charge of routing the information to the MQTT broker.

Nowadays, there are many tools that let you simulate MQTT clients without using any hardware. You need only establish the communication between the MQTT broker and the MQTT client! Below you will find free tools for simulating MQTT communication.

A Free broker to test with you application

- test.mosquitto.org
  Port: 1883

- mqtt.groov.com
  Port: 1883

- broker.hivemq.com
  Port: 1883

A Free Subcriber/Publisher apps to test with you application

- "MQTTBox" Chrom web App
  https://chrome.google.com/webstore/detail/mqttbox/kaajoficamnjijhkeomgfljpicifbkaf

- "MQTTLens" - Chrom web App
  https://chrome.google.com/webstore/detail/mqttlens/hemojaaeigabkbcookmlgmdigohjobjm

- MQTT.fx
  https://mqttfx.jensd.de/index.php/download

2 FUNCTION BLOCKS

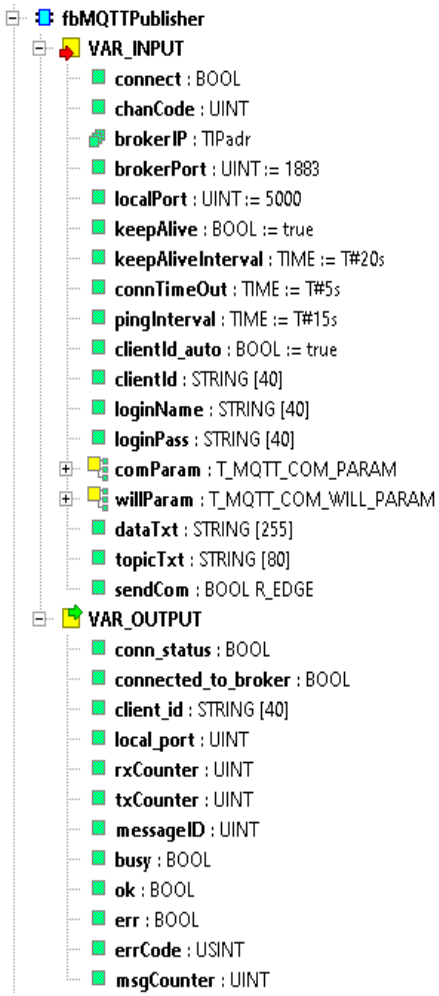
Library: MQTTLib

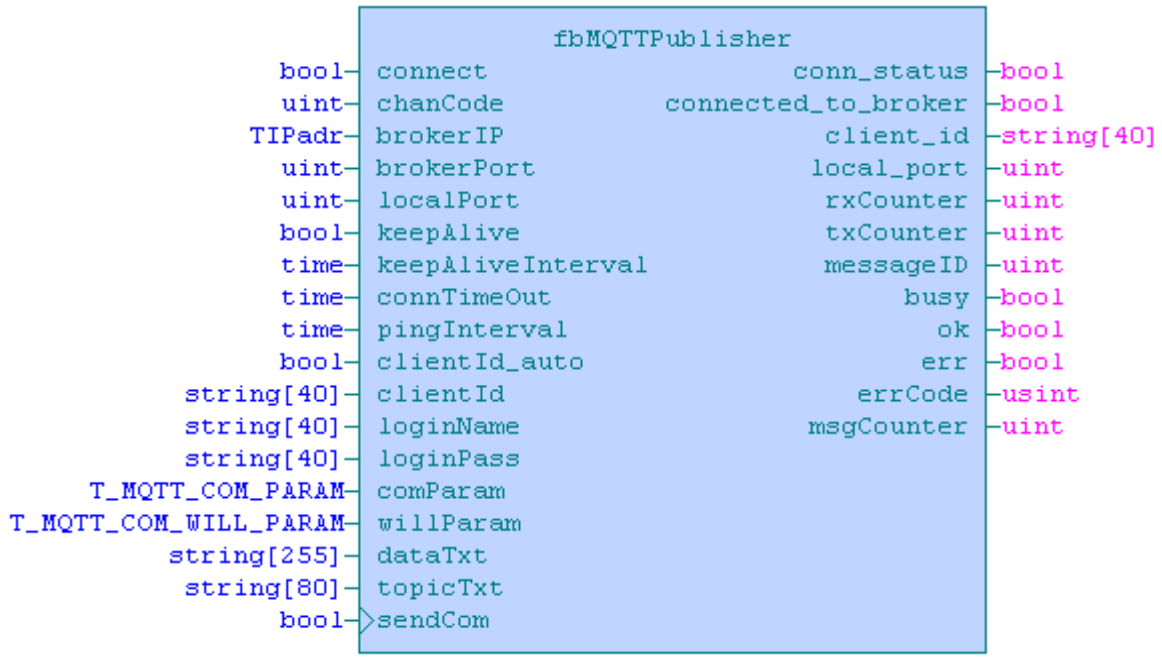
MQTTLib library is supplied as part of Mosaic programming environment. The library contains functions and function blocks provide the ability to create communication between broker and PLC user application.

Functional block	*Description*
fbMQTTPublisher	FB client sending a data to the broker, using a topic name.
fbMQTTSubscriber	FB client tells the broker which topics interest it. Once subscribed, the broker sends messages published to that topic. (In some configurations the broker sends “missed” messages.) A client can subscribe to multiple topics.
fbMQTTPublisherEx	Same as „*fbMQTTPublisher*“ designed for long outgoing MQTT messages, up to 1200 bytes
fbMQTTSubscriberEx	Same as „*fbMQTTSubscriber*“ designed for long incoming MQTT messages, up to 2048 bytes

2.1 Functional block fbMQTTPublisher

Library: MQTTLib





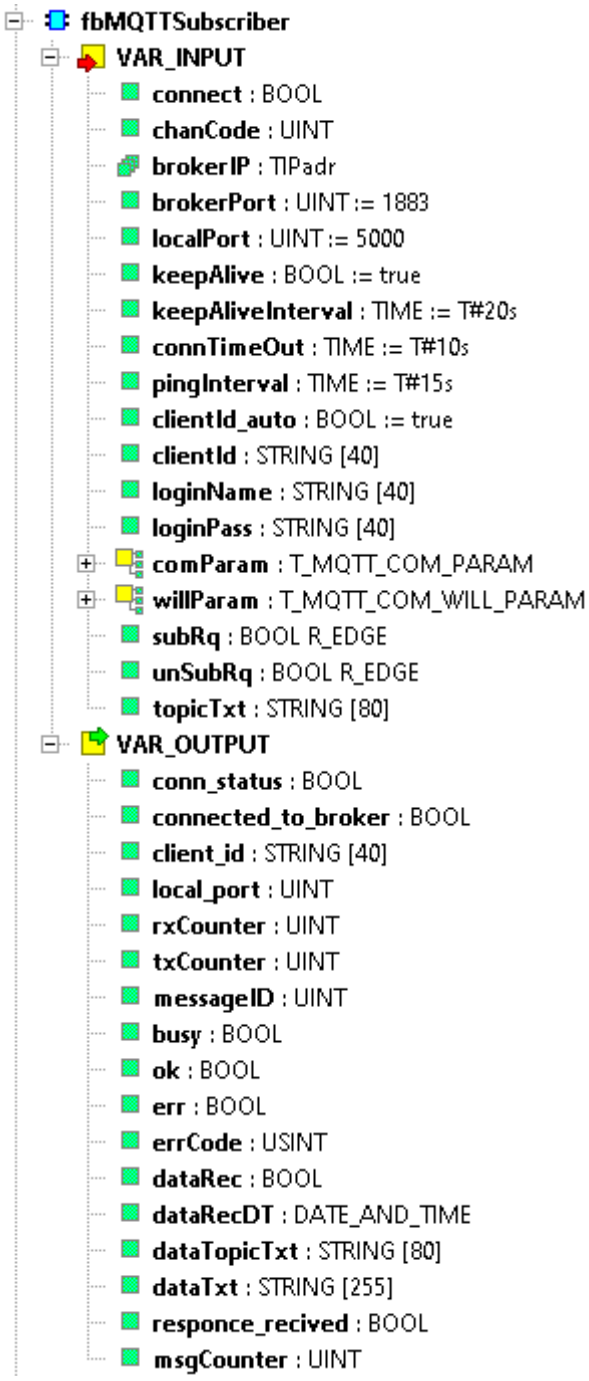
fbMQTTPublisher Variable description:

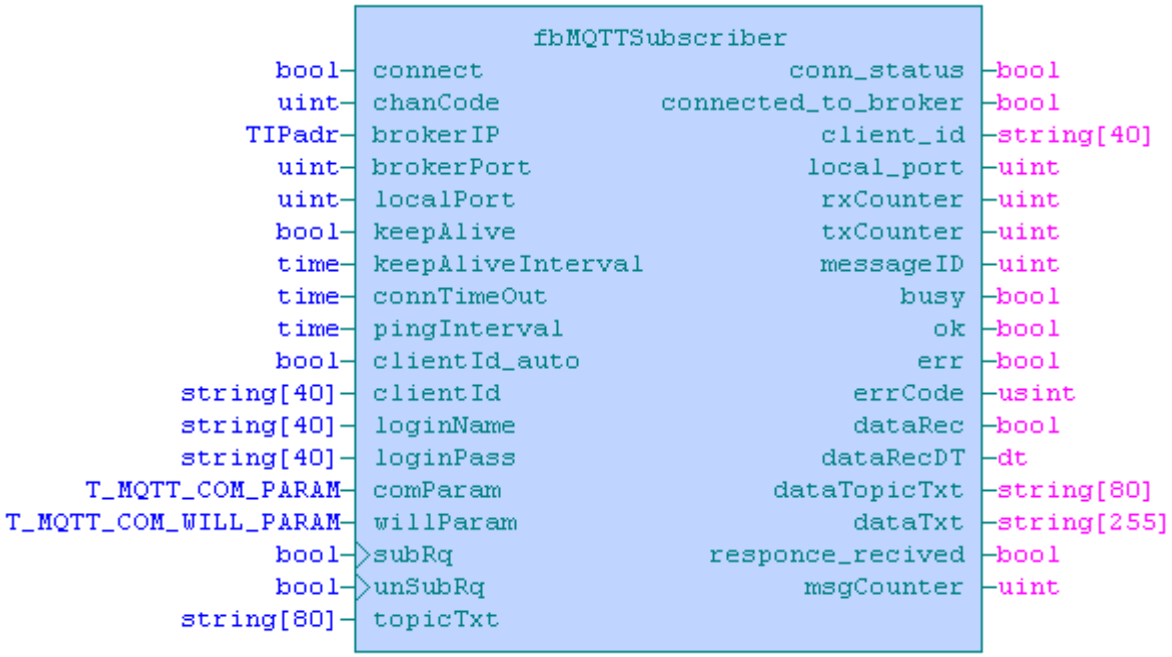
	Variable	Type	Description
VAR_INPUT			
	connect	bool	Connect or disconnect command
	chanCode	uint	Communication channel for MQTT commands and states
	brokerIP	TIPadr	IP address of MQTT broker
	brokerPort	uint	Port of MQTT broker (default value is 1883)
	localPort	uint	PLC local port (default value is 6000)
	keepAlive	bool	FB keep comm channel opened during communication session (Broker keeps all settings)
	keepAliveInterval	time	Maximum time when broker close session and clear all settings of publisher
	connTimeOut	time	Maximum timeout of response from broker
	pingInterval	time	Interval for keep alive comm session between publisher and broker MUST BE: keepAliveInterval > pingInterval
	clientId_auto	bool	Client ID will be generated automatically and append timestamp to MQTT client id
	clientId	string	Static client ID, relevant when clientId_auto = false IMPORTANT: Maximum length is 32 chars
	loginName	string	Log in name, using when required authorization IMPORTANT: Maximum length is 32 chars

	Variable	Type	Description
	loginPass	string	Password, using when required authorization IMPORTANT: Maximum length is 32 chars
	com_param	T_MQTT_COM_PARAM	Parameters of MQTT session
	willParam	T_MQTT_COM_WILL_PARAM	Configuration of parameters for monitoring the connection status between the broker and the MQTT node
	dataTxt	string	Data to transmit IMPORTANT: Maximum length is 255 chars
	topicTxt	string	Topic where published sends data IMPORTANT: Maximum length is 80 chars
	sendCom	bool R_EDGE	Command to send data
VAR_IN_OUT			
VAR_OUT			
	conn_status	bool	Connection status (TCP channel)
	connected_to_broker	bool	Connection state to MQTT broker
	client_id	String	Client ID used in communication session
	local_port	uint	Current PLC local port
	rxCounter	uint	incoming messages counter
	txCounter	uint	Outgoing messages counter
	messageID	uint	message id, used when QOS > 0
	busy	bool	Broadcasting data state
	ok	bool	Ready to broadcast new data
	err	bool	Error occurred
	errCode	usint	Code of error
	msgCounter	uint	Count of outgoing mqtt messages

2.2 Functional block fbMQTTSubscriber

Library: MQTTLib





fbMQTTSubscriber Variable description:

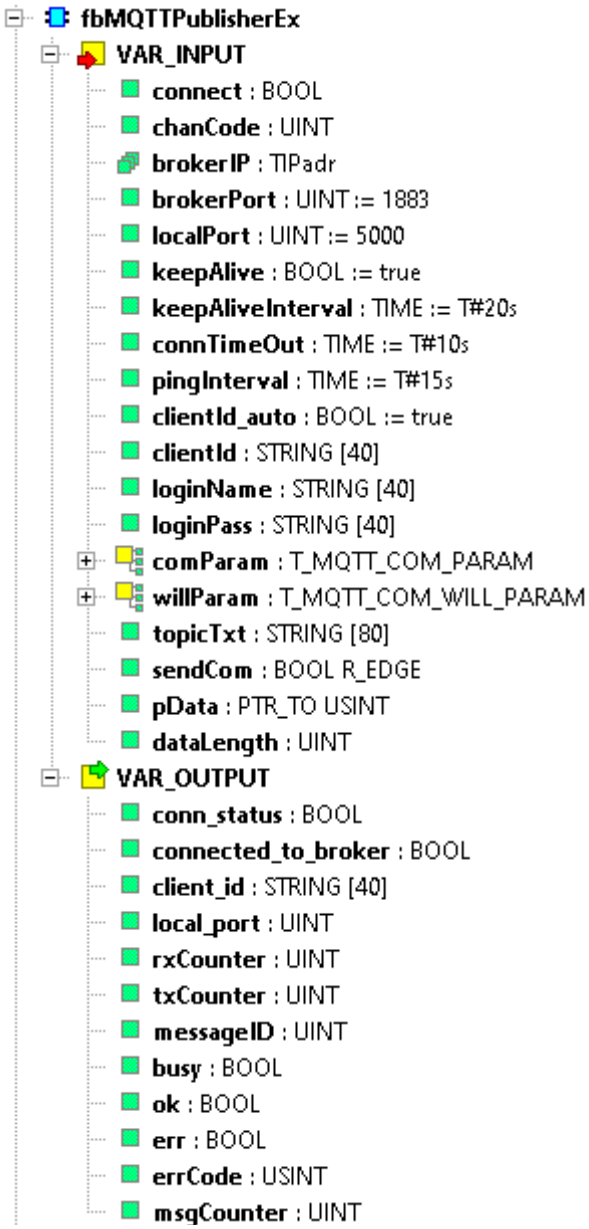
	Variable	Type	Description
VAR_INPUT			
	connect	bool	Connect or disconnect command
	chanCode	uint	Communication channel for MQTT commands and states
	brokerIP	TIPadr	IP address of MQTT broker
	brokerPort	uint	Port of MQTT broker (default value is 1883)
	localPort	uint	PLC local port (default value is 5000)
	keepAlive	bool	FB keep comm channel opened during communication session (Broker keeps all settings)
	keepAliveInterval	time	Maximum time when broker close session and clear all settings of subscriber
	connTimeOut	time	Maximum timeout of response from broker
	pingInterval	time	Interval for keep alive comm session between publisher and broker MUST BE: keepAliveInterval > pingInterval
	clientId_auto	bool	Client ID will be generated automatically and append timestamp to MQTT client id
	clientId	string	Static client ID, relevant when clientId_auto = false IMPORTANT: Maximum length is 32 chars
	loginName	string	Log in name, using when required authorization IMPORTANT: Maximum length is 32 chars

	Variable	Type	Description
	loginPass	string	Password, using when required authorization IMPORTANT: Maximum length is 32 chars
	com_param	T_MQTT_COM_PARAM	Parameters of MQTT session
	willParam	T_MQTT_COM_WILL_PARAM	Configuration of parameters for monitoring the connection status between the broker and the MQTT node
	subRq	bool	Subscribe request
	unSubRq	bool	Unsubscribe request
	subTopicTxt	string	Topic to subscribe or unsubscribe IMPORTANT: Maximum length is 80 chars
VAR_IN_OUT			
VAR_OUT			
	conn_status	bool	Connection status (TCP channel)
	connected_to_broker	bool	Connection state to MQTT broker
	client_id	String	Client ID used in communication session
	local_port	uint	Current PLC local port
	rxCounter	uint	incoming messages counter
	txCounter	uint	Outgoing messages counter
	messageID	uint	Message id, used when QOS > 0
	busy	bool	Broadcasting data state
	ok	bool	Ready to broadcast new data
	err	bool	Error occurred
	errCode	usint	Code of error
	dataRec	bool	Data received
	dataRecDT	DT	Date and time of last received data

	Variable	Type	Description
	dataTopicTxt	string	Received topic
	dataTxt	string	Received data
	responce_recived	bool	Response of last command received
	msgCounter	uint	Counter of incoming mqtt data messages

2.3 Functional block fbMQTTPublisherEx

Library : MQTTLib



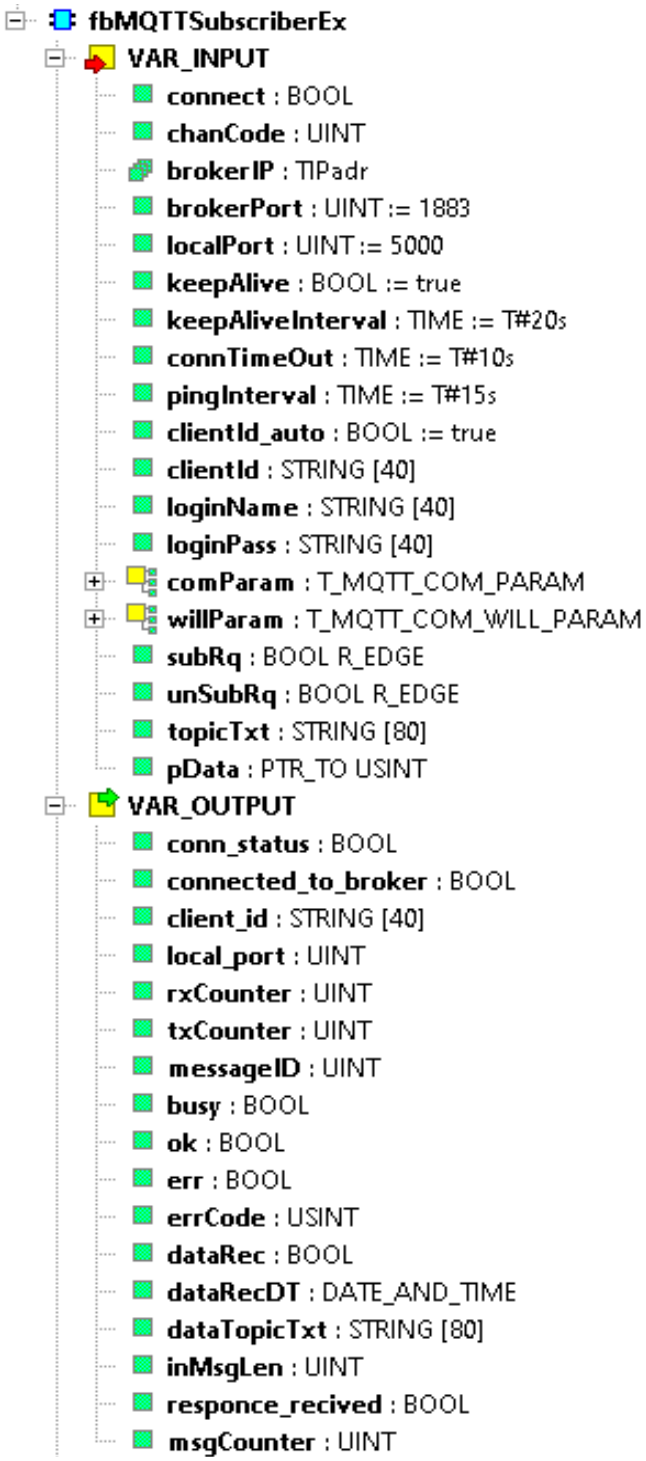
fbMQTTPublisherEx Variable description:

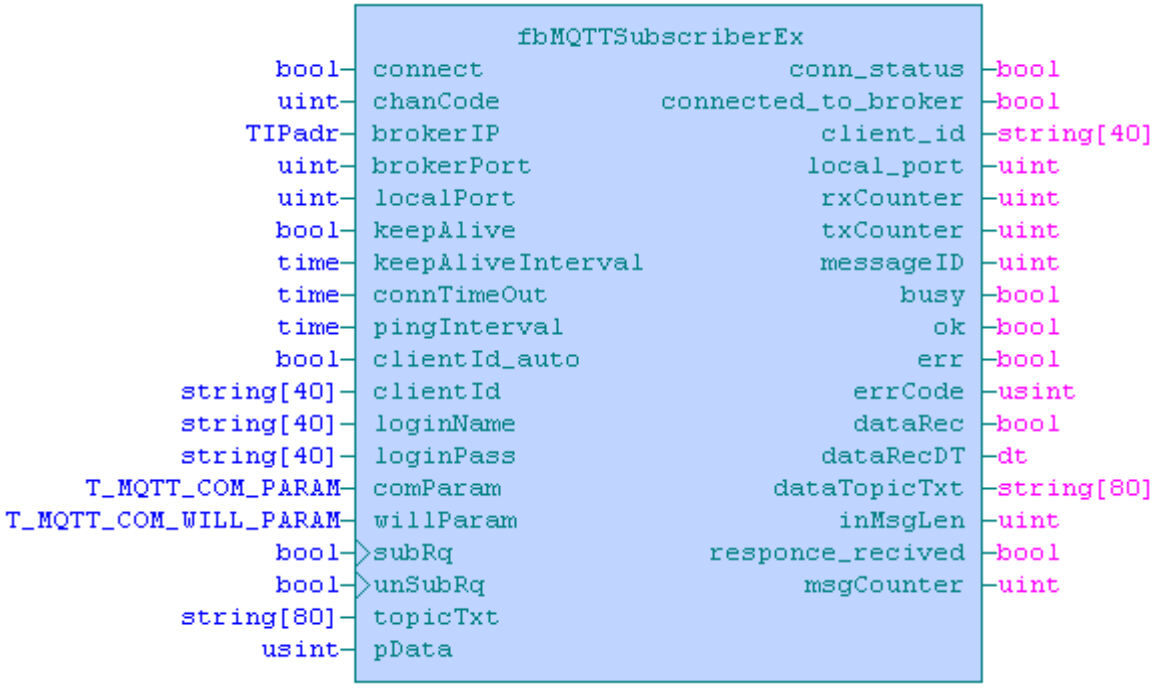
	Variable	Type	Description
VAR_INPUT			
	connect	bool	Connect or disconnect command
	chanCode	uint	Communication channel for MQTT commands and states
	brokerIP	TIPadr	IP address of MQTT broker
	brokerPort	uint	Port of MQTT broker (default value is 1883)
	localPort	uint	PLC local port (default value is 6000)
	keepAlive	bool	FB keep comm channel opened during communication session (Broker keeps all settings)
	keepAliveInterval	time	Maximum time when broker close session and clear all settings of publisher
	connTimeOut	time	Maximum timeout of response from broker
	pingInterval	time	Interval for keep alive comm session between publisher and broker MUST BE: keepAliveInterval > pingInterval
	clientId_auto	bool	Client ID will be generated automatically and append timestamp to MQTT client id
	clientId	string	Static client ID, relevant when clientId_auto = false IMPORTANT: Maximum length is 32 chars
	loginName	string	Log in name, using when required authorization IMPORTANT: Maximum length is 32 chars
	loginPass	string	Password, using when required authorization

	Variable	Type	Description
			IMPORTANT: Maximum length is 32 chars
	com_param	T_MQTT_COM_PARAM	Parameters of MQTT session
	willParam	T_MQTT_COM_WILL_PARAM	Configuration of parameters for monitoring the connection status between the broker and the MQTT node
	topicTxt	string	Topic where published sends data IMPORTANT: Maximum length is 80 chars
	sendCom	Bool R_EDGE	Command to send data
	pData	PTR_TO usint	Pointer to data outgoing buffer IMPORTANT: Maximum length is 512 chars
	dataLength	UINT	Length of outgoing message
VAR_IN_OUT			
VAR_OUT			
	conn_status	bool	Connection status (TCP channel)
	connected_to_broker	bool	Connection state to MQTT broker
	client_id	String	Client ID used in communication session
	local_port	uint	Current PLC local port
	rxCounter	uint	incoming messages counter
	txCounter	uint	Outgoing messages counter
	messageID	uint	message id, used when QOS > 0
	busy	bool	Broadcasting data state
	ok	bool	Ready to broadcast new data
	err	bool	Error occurred
	errCode	usint	Code of error
	msgCounter	uint	Count of outgoing mqtt messages

2.4 Functional block fbMQTTSubscriberEx

Library : MQTTLib





fbMQTTSubscriberEx Variable description:

	Variable	Type	Description
VAR_INPUT			
	connect	bool	Connect or disconnect command
	chanCode	uint	Communication channel for MQTT commands and states
	brokerIP	TIPadr	IP address of MQTT broker
	brokerPort	uint	Port of MQTT broker (default value is 1883)
	localPort	uint	PLC local port (default value is 5000)
	keepAlive	bool	FB keep comm channel opened during communication session (Broker keeps all settings)
	keepAliveInterval	time	Maximum time when broker close session and clear all settings of subscriber
	connTimeOut	time	Maximum timeout of response from broker
	pingInterval	time	Interval for keep alive comm session between publisher and broker MUST BE: keepAliveInterval > pingInterval
	clientId_auto	bool	Client ID will be generated automatically and append timestamp to MQTT client id
	clientId	string	Static client ID, relevant when clientId_auto = false IMPORTANT: Maximum length is 32 chars

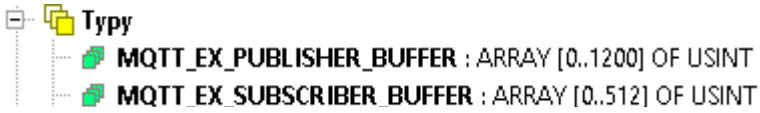
	Variable	Type	Description
	loginName	string	Log in name, using when required authorization IMPORTANT: Maximum length is 32 chars
	loginPass	string	Password, using when required authorization IMPORTANT: Maximum length is 32 chars
	com_param	T_MQTT_COM_PARAM	Parameters of MQTT session
	willParam	T_MQTT_COM_WILL_PARAM	Configuration of parameters for monitoring the connection status between the broker and the MQTT node
	subRq	bool	Subscribe request
	unSubRq	bool	Unsubscribe request
	subTopicTxt	string	Topic to subscribe or unsubscribe IMPORTANT: Maximum length is 80 chars
	pData	PTR_TO USINT	Pointer to MQTT incoming message buffer
VAR_IN_OUT			
VAR_OUT			
	conn_status	bool	Connection staus (TCP channel)
	connected_to_broker	bool	Connection state to MQTT broker
	client_id	String	Client ID used in communication session
	local_port	uint	Current PLC local port
	rxCounter	uint	incomming messages counter
	txCounter	uint	Outgoing messages counter
	messageID	uint	Message id, used when QOS > 0
	busy	bool	Broadcasting data state
	ok	bool	Ready to broadcast new data
	err	bool	Error occurred
	errCode	usint	Code of error

	Variable	Type	Description
	dataRec	bool	Data received
	dataRecDT	DT	Date and time of last received data
	dataTopicTxt	string	Received topic
	dataTxt	string	Received data
	responce_recived	bool	Response of last command received
	msgCounter	uint	Counter of incoming mqtt data messages

3 DATA TYPES

Library: MQTTLib

MQTTLib library defines the following types of variables:

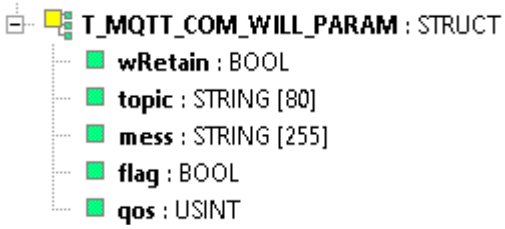


Name	*Type*	*Description*
MQTT_EX_PUBLISHER_BUFFER	USINT	Outgoing data buffer for fbMQTTPublisherEx
MQTT_EX_SUBSCRIBER_BUFFER	USINT	Incoming data buffer for fbMQTTSubscriberEx

Name	*Type*	*Description*
T_MQTT_COM_PARAM	struct	Configuration of MQTT comm. sessions

T_MQTT_COM_PARAM

Variable	*Type*	*Description*
pRetain	bool	This flag indicates whether the broker will retain the message as the last known message for this topic
qos	usint	Quality of Service (value:0,1,2)
dup	bool	If the DUP flag is set to 0, it indicates that this is the first occasion that the Client or Server has attempted to send this MQTT PUBLISH Packet. If the DUP flag is set to 1, it indicates that this might be re-delivery of an earlier attempt to send the Packet.
clean	bool	This bit specifies the handling of the Session state. If CleanSession is set to 0, the Server MUST resume communications with the Client based on state from the current Session (as identified by the Client identifier). If there is no Session associated with the Client identifier the Server MUST create a new Session.



Name	Type	Description
T_MQTT_COM_WILL_PARAM	struct	Configuration of parameters for monitoring the connection status between the broker and the MQTT node

Variable	Type	Description
wRetain	bool	This bit specifies if the Will Message is to be Retained when it is published
topic	usint	If client disconnects unexpectedly, the broker publishes to this message with the payload Will Message
mess	string	If the DUP flag is set to 0, it indicates that this is the first occasion that the Client or Server has attempted to send this MQTT PUBLISH Packet. If the DUP flag is set to 1, it indicates that this might be re-delivery of an earlier attempt to send the Packet.
flag	bool	If the Will Flag is set to 1 this indicates that, if the Connect request is accepted, a Will Message MUST be stored on the Server and associated with the Network Connection.
qos	usint	PUBLISH Quality of Service for Will Message (value:0,1,2)

4 CONSTANTS

Library: MQTTLib

Name	*Type*	*Description*
MQTT_CONTROL_ERROR_CODE_OK	USINT	0 - Ok, without errors
MQTT_CONTROL_ERROR_CODE_BROKER_ANSWER_TIME_OUT	USINT	201 – Broker response timeout
MQTT_CONTROL_ERROR_CODE_SUBSCRIBE_FAILED	USINT	202 – Subscribe command is failed
MQTT_CONTROL_ERROR_CODE_BROKER_DISCONNECTED	USINT	203 – Broker is disconnected
MQTT_CONTROL_ERROR_CODE_ANSWER_LENGTH_TO_LONG	USINT	204- Incoming data is too long
MQTT_CONTROL_ERROR_CODE_LOGIN_FAILED	USINT	205 – Login to broker is failed (name or password is incorrect)
MQTT_CONTROL_ERROR_CODE_BUFFER_OVERFLOW	USINT	206- Receive buffer overflow error
MQTT_CONTROL_ERROR_CODE_DATA_PROCESSING_ERROR	USINT	207- Error processing received message
MQTT_CONTROL_ERROR_CODE_PUBLISHER_PAYLOAD_OUT_OF_RANGE	USINT	208-Transmission data length exceeded

NOTE: In case when error from 1 up to 64, please refer to ComLib errors.

5 GLOBAL VARIABLES

Library: MQTTLib

The library MQTTLib, hasn't global variables.

6 FUNCTIONS

Library: MQTTLib

The library MQTTLib, hasn't additional functions.

7 COMMUNICATION CHANNEL SETTINGS

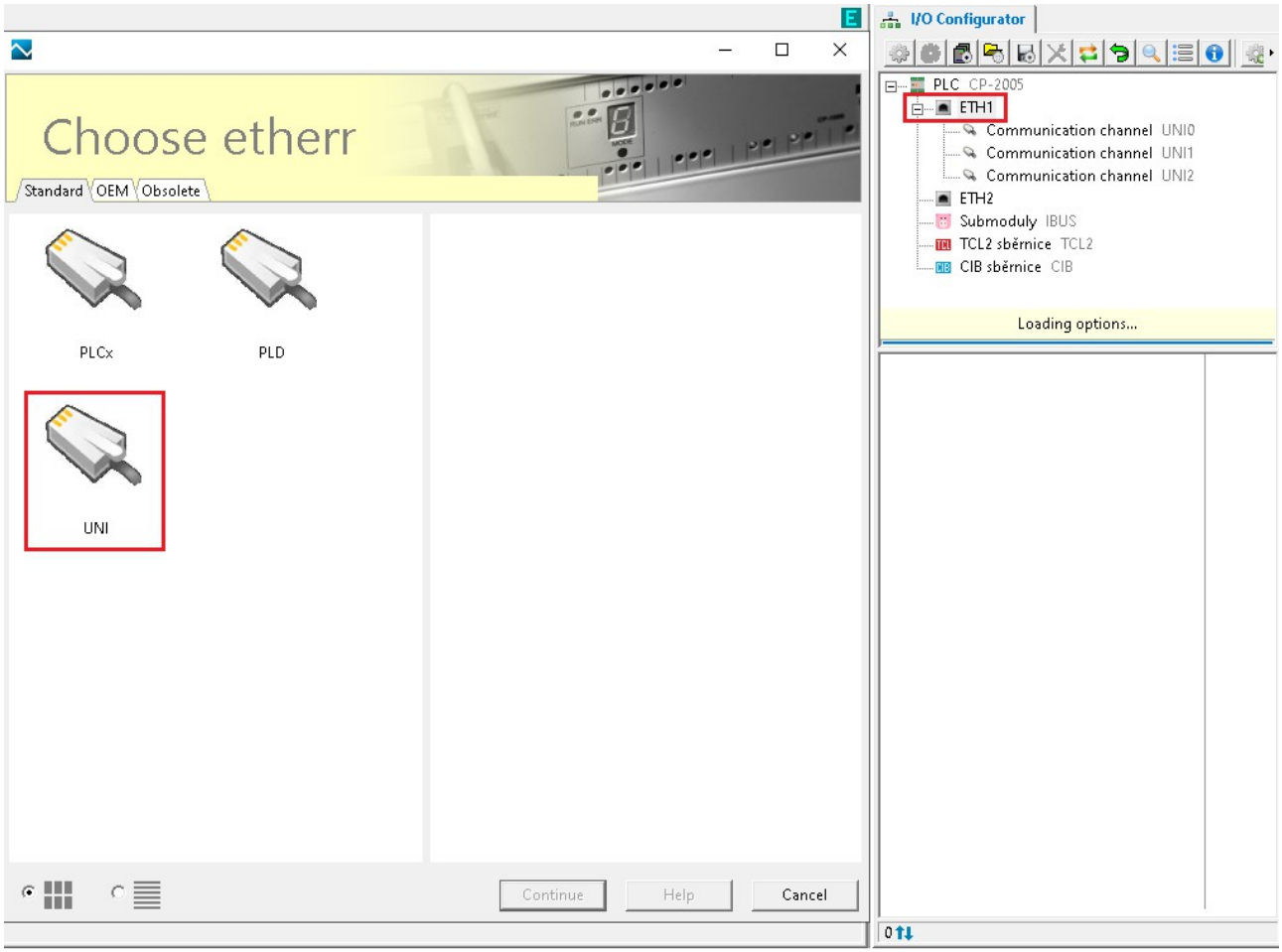
Library: MQTTLib

MQTT Library require channels set to UNI mode. This setting can be made in the I/O configurator

7.1 Ethernet channel setup

Library: MQTTLib

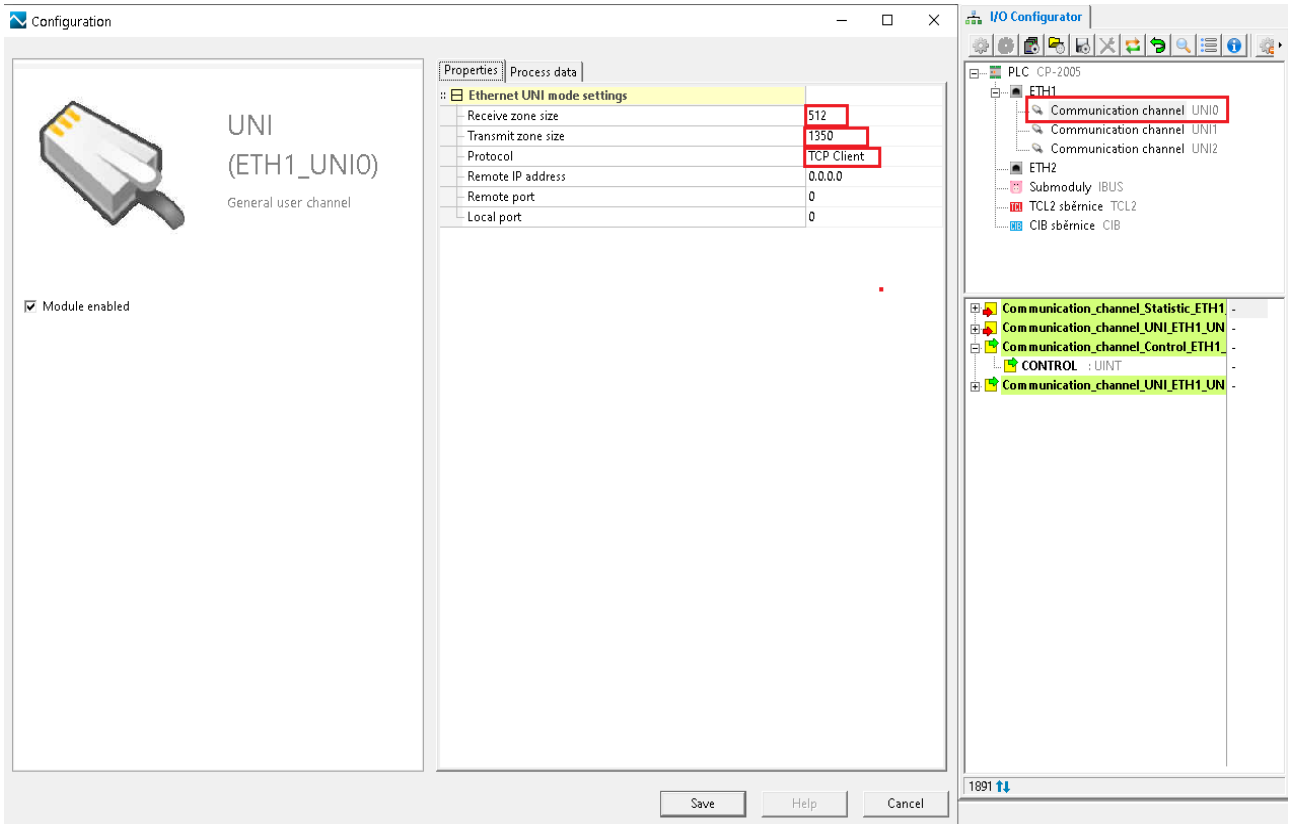
Open channel parameters setting in I/O configuration (ETH1). Choose **UNI** channel in setting dialogue and press on 'Save' button. See picture below.



7.2 Unsecured communication for Ethernet channel settings

To configure created or exist communication channel, press double click on left mouse button. In opened configuration dialogue, set following parameters. See picture below.

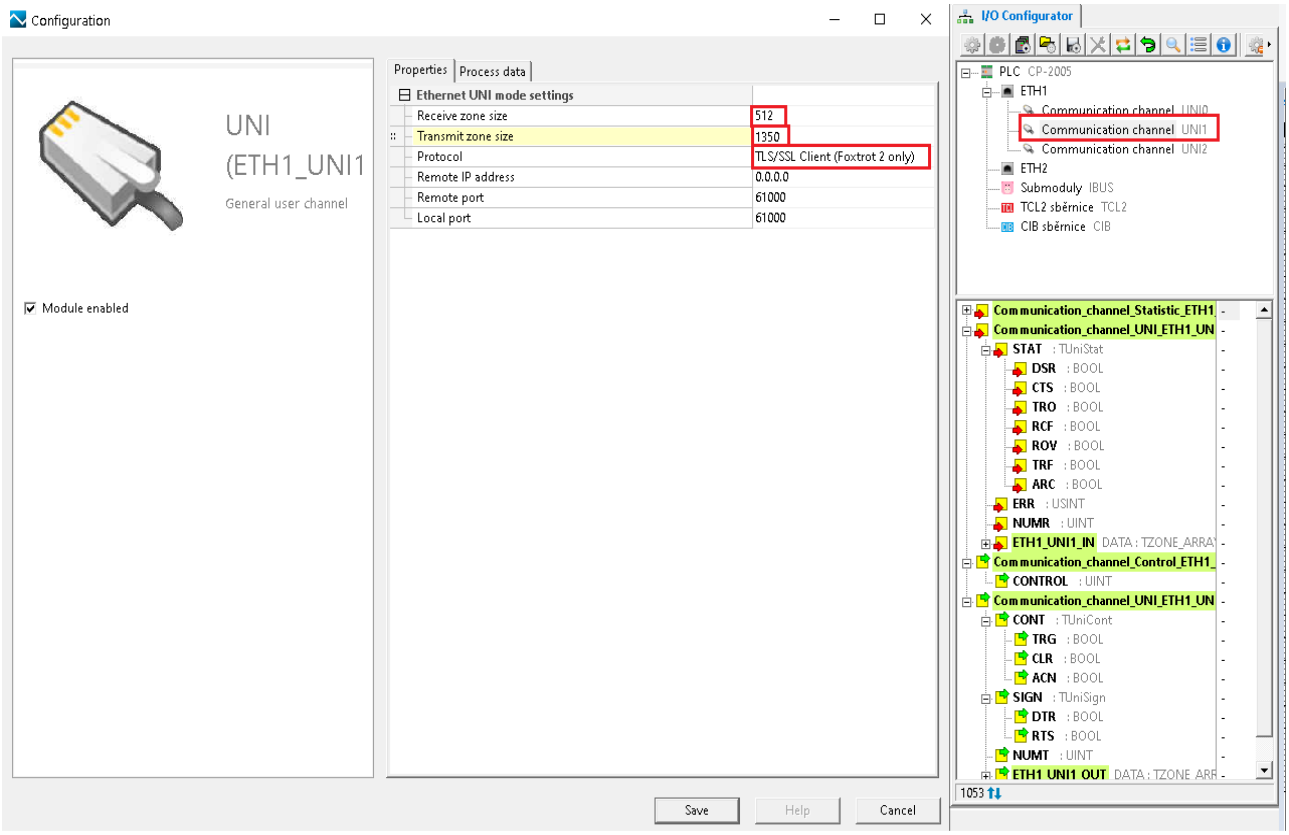
- Receiving zone 512 bytes,
- Transmit zone size 1350 bytes
- Protocol type, TCP client
- Remote IP address is 0.0.0.0
- Remote port 0
- Local port 0



7.3 Secure communication for Ethernet channel setting (TLS/SSL)

To configure created or exist communication channel, press double click on left mouse button. In opened configuration dialogue, set following parameters. See picture below.

- Receiving zone 512 bytes,
- Transmit zone size 512 bytes
- Protocol type, TLS/SSL client
- Remote IP address is 0.0.0.0
- Remote port 0
- Local port 0



8 EXAMPLES

Example 1. MQTT Publisher
In „Main“ program defined one functional block and two control structures:
In this specific example used IP address of mqtt broker installed PC. To establish connection with remote broker. please use FB NsLookUpEx in InternetLib

```
PROGRAM prgMain
  VAR
    MQTTControl2          : fbMQTTPublisher;
    brokerIPaddr          : STRING := '127.0.0.1';//IP of local broker
    remotePort            : UINT := 1883;
    localPort             : UINT := 60000;
    keepAlive             : bool := true;
    keepAliveInterval     : time := T#60s;
    pingInterval          : time := T#10s;
    connTimeOut           : time := T#10s;
    connect               : bool := true; // connect command for publisher
    loginName             : string[40] := 'test';
    loginPass             : string[40] := 'test';
    deviceID              : string[40] := 'TEST_MQTT_LIB_PUB';
    sendCom               : bool;//manual command to send data to broker
    pubDataTxt            : string[255] := '"{"light_1": "on","temp_1": 24.2}"';
    pubTopicTxt           : string[80]  := 'house/room1';

   // Optional parameters
    com_param             : T_MQTT_COM_PARAM;
    willParam             : T_MQTT_COM_WILL_PARAM := ( topic :=
'house/connect/devices', mess := '{"device":"sensor 234", "connection":
"FALSE"}');


  END_VAR
  VAR_TEMP
  END_VAR

    MQTTControl2(chanCode          := ETH1_UNI0,
                 connect           := connect,
                 sendCom           := sendCom,
                 brokerIP          := STRING_TO_IPADR(brokerIPaddr),
                 brokerPort        := remotePort,
                 localPort         := localPort,
                 keepAlive         := keepAlive,
                 keepAliveInterval := keepAliveInterval,
                 clientId_auto     := FALSE,
                 pingInterval      := pingInterval,
                 connTimeOut       := connTimeOut,
                 com_param         := com_param,
                 willParam         := willParam,
                 clientId          := deviceID,
                 loginName         := loginName,
                 loginPass         := loginPass,
                 dataTxt           := pubDataTxt,
                 topicTxt          := pubTopicTxt
                );


END_PROGRAM
```

Example 2. Subscriber
In „Main“ program defined one functional block and two control structures:
In this specific example used IP address of mqtt broker installed PC. To establish connection with remote broker. please use FB NsLookUpEx in InternetLib

```
PROGRAM prgMain
  VAR
    MQTTControlSubs       : fbMQTTSubscriber;
    brokerIPaddr          : STRING := '127.0.0.1';//IP of local broker
    remotePort            : UINT := 1883;
    pingInterval          : time := T#10s;
    connectSubs           : bool := true; // connect command for publisher

    loginName             : string[32] := 'test';
    loginPass             : string[32] := 'test';
    deviceID              : string[40] := 'TEST_MQTT_LIB_SUB';

    connTimeOutSubs       : time := T#10s;
    connectSubs           : bool := true;
    localPortSubs         : UINT := 50000;
    keepAliveSubs         : bool := true;
    keepAliveIntervalSubs : time := T#60s;
    com_param             : T_MQTT_COM_PARAM;
    willParam             : T_MQTT_COM_WILL_PARAM := ( topic := 'gateway/con-
nect/devices', mess := '{"device":"pannel 51", "connection": "FALSE"}');


    subTopicTxt           : string[80]  := 'house/room1';

    inDataTxt             : string[255];
    inDataTopicTxt        : string[80];
    inDataTime            : DT;
  END_VAR
  VAR_TEMP
  END_VAR



    MQTTControlSubs(chanCode          := ETH1_UNI1,
                    brokerIP          := STRING_TO_IPADR(brokerIPaddr),
                    brokerPort        := remotePort,
                    localPort         := localPortSubs,
                    connect           := connectSubs,
                    keepAlive         := keepAliveSubs,
                    keepAliveInterval := keepAliveIntervalSubs,
                    pingInterval      := pingInterval,
                    connTimeOut       := connTimeOutSubs,
                    com_param         := com_param,
                    willParam         := willParam,
                    clientId          := deviceID,
                    loginName         := loginName,
                    loginPass         := loginPass,
                    subTopicTxt       := subTopicTxt

                   );
```

```
    IF MQTTSubs.dataRec THEN // Incomming data

         inDataTxt      := MQTTControlSubs.dataTxt;
         inDataTopicTxt := MQTTControlSubs.dataTopicTxt
         inDataTime     := MQTTControlSubs.dataRecDT;
    END_IF;
END_PROGRAM
```

Teco a.s. Průmyslová zóna Šťáralka 984, 280 02 Kolín, tel. +420 321 401 111,
e-mail: teco@tecomat.cz