OMRON

MQTT

User Manual

O085-E-01

MQTT Interface on Power PMAC

Power PMAC can support the MQTT (Message Queue Telemetry Transport) communications protocol. MQTT is a lightweight publish/subscribe network protocol. Power PMAC acts as a network client that can publish messages to the network broker (server) and/or subscribe to messages from the broker, which originated from other clients on the network.

Power PMAC's MQTT implementation communicates over a TCP/IP Ethernet connection using the primary Ethernet port. It operates as a Linux OS application running independently from the Power PMAC machine-control firmware, but communicating with it.

Power PMAC shares data with the network through its global (P) variables. It can publish the value of any P-variable to the broker, or subscribe to a topic whose value it places in any P-variable. Power PMAC has 65,536 P-variables (P0 – P65535), which can be accessed directly (e.g. P100), through a **`#define`** substitution, or a declared **`global`** variable name.

The Power PMAC client application can subscribe to up to 1000 variables. An array of consecutively numbered P-variables for publishing or subscription can be up to 1000 variables in length.

Enabling and Connecting to the MQTT Broker

The MQTT interface must first be enabled by setting saved setup element **Mqtt.Enable** to 1 to start the MQTT application in Power PMAC. If this value has been saved to flash memory, it will be enabled automatically on power-on/reset. At the start of the enabling process, **MqttEnable** is set to 0, and then on a successful enabling, it is set back to 1. This element must be equal to 1 before any other actions can be taken on the network.

Login Information

If the broker requires a user name and possibly a password for the connection, the **`mqttsetlogin`** function must be called to establish the user name and (optional) password that will be used in the connection to the broker. Of course, these must match what the broker is expecting for the connection. An example is:

```
MqttResult = mqttsetlogin("PowerPMAC1, 314159");
```

Communications Security

At connection time, the security features of the connection, if any, are established. If **Mqtt.UseTLS** is set to 1, a secure connection will be attempted using Transport Layer Security (TLS) functionality.

A secure connection requires both "certificates" of the client and the broker, so they can verify their identity to the other party, and "keys" for encryption and decryption of the data sent. Refer to the description of **Mqtt.UseTLS** in the Software Reference Manual for details of implementing the certificates and keys.

Establishing the Connection

The next step is to connect the Power PMAC client to the network broker using the **`mqttconnect`** function. Its argument is the IP address of the broker on the Ethernet network.

This is typically expressed as a text string in quotes, e.g. **"192.168.0.215"**. A "hostname" alias can also be used – see the description of the **mqttconnect** function for details.

The port number to be used on the broker's Ethernet interface is specified by the value of **Mqtt.Port** when this function is called. This is usually 1883 if unsecured communication is desired, or 8883 if secure communication is desired.

The "keep alive" time for the connection is specified by the value of saved setup element **Mqtt.KeepAliveTime** when this function is called.

At connection time, the client ID of the Power PMAC on the broker's network is established. If **Mqtt.ClientID** is set to a value greater than zero, this value will be used as Power PMAC's client ID. If **Mqtt.ClientID** is set to zero, the broker will automatically assign a client ID number to the Power PMAC.

The connecting function returns a value of 0 on a successful connection, or nan (not-a-number) on an unsuccessful connection. Power PMAC sets status element **Mqtt.Connected** to 1 on a successful connection.

Subscribing to Topics from the Broker

Once enabled and connected, the Power PMAC MQTT interface can subscribe to topics using the **mqttsub** function.

When Power PMAC has subscribed to an MQTT topic, every time the broker receives a new published value on the topic, it will automatically send a message with this value to the Power PMAC, which will store that value in the specified global (P) variable. It is not necessary to use the **mqttsub** function multiple times for a single topic.

The name used for the variable in the function can be the actual P-variable (e.g. **P50**, **P(MyBase+10)**) or a declared/defined alias. Whichever of these is used to specify the variable in the function, it must match the topic name used in the broker.

For example, if you use the Power PMAC definition **#define SolderTemp P50** in the IDE project, and then call the function **mqttsub(SolderTemp)**, the Power PMAC will subscribe to the broker's **SolderTemp** topic, and when it receives a value from the broker, it will put this value into global variable **P50**. Because of the definition in the IDE project, other Power PMAC functions can use the defined name.

Similarly, if you use the Power PMAC declaration **global DoorOpen** in the IDE project, and then call the function **mqttsub(DoorOpen)**, the Power PMAC will subscribe to the broker's **DoorOpen** topic, and when it receives a value from the broker, it will put this value into the P-variable that has been automatically assigned to the **DoorOpen** alias by the IDE project manager. Other Power PMAC tasks can then use the **DoorOpen** variable alias; it is not necessary to know the underlying P-variable name.

It is possible to subscribe to an array of consecutively numbered variables with a single **mqttsub** function. For example, **mqttsub(100,P1000)** causes a subscription to 100 variables that will be placed in Power PMAC variables P1000 through P1099. It is strongly

recommended that the actual P-variable names be used here instead of a defined or declared alias.

Publishing Topics to the Broker

Power PMAC can also publish topics to the MQTT broker using the **mqttpub** function. The arguments for this function specify the global (P) variable whose value is to be published, and optionally the number of consecutively numbered variables to be published (if more than 1).

The name used for the variable in the function can be the actual P-variable (e.g. **P100**, **P(MyBase+20)**) or a declared/defined alias. Whichever of these is used to specify the variable in the function, this determines the topic name used in the broker. Other clients must use this name to subscribe to the topic.

For example, if you use the Power PMAC definition **#define CycleCount P100**, and then call the function **mqttpub(CycleCount)**, the Power PMAC will publish to the broker's **CycleCount** topic, sending the value of P100 to the broker. Because of the definition in the IDE project, other Power PMAC functions can use the defined name.

It is possible to publish an array of consecutively numbered variables with a single **mqttpub** subscription. For example, **mqttpub(50,P8192)** causes Power PMAC to publish 50 variable valuess from Power PMAC variables P8192 through P8241. It is strongly recommended that the actual P-variable names be used here instead of a defined or declared alias.

If you want to publish a value that is not already in a Power PMAC P-variable, you must first copy that value into a P-variable. For example:

```
global CS1InPosition;
...
CS1InPosition = Coord[1].InPos;
mqttpub(CS1InPosition);
```

The **mqttpub** function needs to be called each time Power PMAC publishes a value to the broker.

Properties of the Message Transmissions

There a several aspects as to how messages are transmitted between Power PMAC as an MQTT client and the MQTT broker. These are controlled by a set of saved setup elements whose values are used at the time a message is sent.

Mqtt.QoS determines the "quality of service" of the message transmission. It can be set to a value of 0, 1, or 2, with the higher values specifying more handshaking and acknowledgement for the message.

Mqtt.Retain tells the broker what to do with the message that Power PMAC publishes if no client is presently subscribing to that topic. If it is set to 0, the broker will discard the message. If it is set to 1, the broker will retain the message for a possible future subscriber.

Mqtt.Prefix, if set to a value greater than 0, specifies the numerical prefix to be appended to the start of a message published by Power PMAC. If it is set to 0, no prefix is used. Prefixes are

commonly used to distinguish otherwise identical topics published by multiple clients on the network.

MQTT Status Files

The **/var/log** directory contains the following MQTT status files:

mqtt_status.txt – contains a table of all of the **Mqtt** data structure status elements and saved setup elements as well as the port number, broker hostname/IP address, and Username.

mqtt_subs.txt – contains a list of all of the topics that the MQTT client is currently subscribed to.

Closing the Connection

Power PMAC's connection to the MQTT broker can be ended at any time using the **mqttclose()** function.

Examples

These code examples in the Power PMAC script environment demonstrate the use of Power PMAC as an MQTT client.

Example 1

This example PLC program uses MQTT to control the speed of a motion program.

global definitions.pmh

```
global result;
global count = 0;
global init = 0;
global speed = 0;
global old_speed = 0;
&1#1->x
```

plc1.plc

```
open plc 1

speed = old_speed;                // To initialize
mqtt.port = 1883;
mqtt.enable = 1;
while (mqtt.enable == 0) {}       // wait until client app starts
result = mqttconnect("192.168.0.215");  // connect to MQTT broker

while (mqtt.connected == 0) {}    // wait until connected before proceeding
result = mqttsub(speed);          // subscribe to receive updates to "speed"

while (speed == old_speed) {}     // wait until an MQTT client pubs a new speed
enable 1;                         // enable coordinate system 1
start 1:1;                        // start prog 1 in CS 1
while (coord[1].progactive) {}    // wait until prog 1 is done

result = mqttunsub(speed);        // stop receiving updates to "speed"
result = mqttclose();             // close the MQTT connection
disable plc 1;

close
```

prog1.pmc

```
open prog 1

// &1#1->x defined in "global definitions.pmh"
Motor[1].MaxSpeed = speed

inc rapid X5000;

close
```

Example 2

This example PLC program demonstrates using a TLS/SSL connection and subscribing to multiple topics at once.

global definitions.pmh

```
global result;
```

plc2.plc

```
open plc 2

mqtt.usetls = 1;
mqtt.port = 8883;
mqtt.qos = 2;
mqtt.enable = 1;
while (mqtt.enable == 0) {}       // wait until client app starts
result = mqttsetlogin("admin, 1234");
result = mqttconnect("192.168.0.215");  // connect to broker w/ TLS

while (mqtt.connected == 0) {}    // wait until connected before proceeding
result = mqttsub(100, P1);        // subscribe to topics for P1 - P100

while (mqtt.msgsrcvd < 1000) {}   // wait until 1000 messages are received

result = mqttclose();
disable plc 2;

close
```

Example 3

This example PLC program demonstrates publishing the value of global variable as it is updated.

global definitions.pmh

```
global result;
global count;
```

plc3.plc

```
open plc 3

speed = old_speed;

mqtt.qos = 1;
mqtt.retain = 0;
mqtt.port = 1883;
mqtt.enable = 1;
while (mqtt.enable == 0) {}       // wait until client app starts
result = mqttconnect("192.168.0.215");  // connect to MQTT broker

while (mqtt.connected == 0) {}    // wait until connected before proceeding

count = 0;
result = mqttpub(count;)
while (count < 1000)
{
       result = mqttpub(count);          // pub new value for each iteration
       count++;
}

result = mqttclose();

disable plc 3;

close
```

MQTT Interface Saved Setup Elements

These control elements specify details as to how the MQTT client interface on the Power PMAC will work.

Mqtt.ClientID

Description: Power PMAC MQTT client number for broker

Units: Enumeration

Range: 0 .. 32,767

Default: 0 (externally specified)

Mqtt.ClientID specifies the client number of the Power PMAC on the broker's MQTT network. If set to a value greater than 0 when connection to the broker is established with the **`mqttconnect`** function, it specifies the client number to be used.

If set to 0 when connection to the broker is established with the **`mqttconnect`** function, the broker specifies the client number that will be used for the Power PMAC.

Each client on a broker's MQTT network must have a unique client number.

This **Mqtt** element is new in V2.8.1 firmware, released 4th quarter 2024.

Mqtt.Enable

Description: Power PMAC MQTT application enable/disable control

Units: Boolean

Range: 0 .. 1

Default: 0

Mqtt.Enable specifies the desired enable/disable state of Power PMAC's built-in MQTT client application. It must be set to 1 before connection to the MQTT broker can be made with the **`mqttconnect`** function. If the saved value of **Mqtt.Enable** is 1, the MQTT client application is automatically enabled on power-on/reset.

When Power PMAC sets **Mqtt.Enable** to 1, the client application clears it to 0 when it starts the enabling process, and sets it back to 1 on successful completion of the enabling process. **Mqtt.Enable** will remain at 0 if the enabling process is not successful.

This **Mqtt** element is new in V2 8.1 firmware, released 4th quarter 2024.

Mqtt.KeepAliveTime

Description: Power PMAC MQTT client maximum time between packets

Units: Seconds

Range: 0, 5 .. 65,535

Default: 60

Mqtt.KeepAliveTime, if set to a positive value, specifies the maximum time, in seconds, that are allowed to elapse between MQTT protocol packets sent by the Power PMAC acting as a client. This value is sent to the MQTT broker on initial connection, and if this time is exceeded, the broker will "ping" the Power PMAC to verify the connection is still valid.

If **Mqtt.KeepAliveTime** is set to 0, this keep-alive mechanism is disabled.

This **Mqtt** element is new in V2 8.1 firmware, released 4th quarter 2024.

Mqtt.Port

Description: MQTT broker software port number

Units: Enumeration

Range: 0... 32,767

Default: 8883

Mqtt.Port specifies the software port number on the broker to which Power PMAC will address its communications to the broker. This is determined by the configuration of the broker. The port number is usually 1883 for an "insecure" connection, or 8883 for a connection secured by TLS.

This **Mqtt** element is new in V2 8.1 firmware, released 4th quarter 2024.

Mqtt.Prefix

Description: Power PMAC MQTT topic prefix use control

Units: Enumeration

Range: 0... 65,535

Default: 0 (no prefix)

Mqtt.Prefix, if set to a positive number, specifies the value of the prefix that will be prepended to the MQTT topic in messages (e.g. **`55/global/P3`**) published by Power PMAC to the broker. It

is used each time an **mqttpub** function is called, so different prefix values can be used for different topics.

If **Mqtt.Prefix** is set to the default value of 0, no prefix will be added to the MQTT topic (e.g. **global/P3**).

Prefixes are particularly useful if there are multiple Power PMACs on the same MQTT network, permitting the network to distinguish between the same topic published by different PMACs.

This **Mqtt** element is new in V2.8.1 firmware, released 4th quarter 2024.

Mqtt.QoS

Description: Power PMAC MQTT Quality of Service

Units: Enumeration

Range: 0 .. 2

Default: 0

Mqtt.QoS specifies the "Quality of Service" level value used in transferring MQTT messages.

The Quality of Service level value defines the amount of handshaking involved in the transfer.

- QoS level 0 specifies to deliver the message at most once. Power PMAC simply publishes the message, and there is no acknowledgement by the broker. This is the simplest and lowest-overhead method of sending a message.

- QoS level 1 specifies to deliver the message at least once. The broker sends an acknowledgement back to the Power PMAC, but if the acknowledgement is lost, Power PMAC will send the message again, and will repeat this until is receives an acknowledgement. This means that sending is guaranteed, although the message may reach the broker more than once.

- QoS level 2 specifies to deliver the message exactly once. This causes a sequence of four messages between the Power PMAC and the broker, confirming that the message has been sent and the acknowledgement received. When this handshake has been completed, both sender and receiver are sure that the message was sent exactly once. This is the highest level of service.

Mqtt.QoS is used each time Power PMAC publishes a message with the **mqttpub** function, so it is possible to publish different messages with different QoS values. It is also used for receiving subscribed messages, but these messages cannot use a higher QoS level than they were published with to the broker.

This **Mqtt** element is new in V2 8.1 firmware, released 4th quarter 2024.

Mqtt.Retain

Description: Power PMAC MQTT published message retention policy

Units: Enumeration

Range: -1 .. 1

Default: 0

Mqtt.Retain specifies Power PMAC's "retention" policy associated with MQTT messages it publishes to the broker.

The selected retention policy defines what the broker does with the message if no clients are subscribing to that subject.

- A retention value of 0 specifies that if no clients are presently subscribing to that message, the message is discarded by the broker.
- A retention value of 1 specifies that the message is to be retained by the broker, even if no clients are presently subscribing to that message. This permits clients to subscribe in the future and receive this message.
- A retention value of -1 specifies that a previously retained message on this topic from Power PMAC in the broker is to be discarded.

Mqtt.Retain is used each time Power PMAC publishes a message with the **`mqttpub`** function, so it is possible to publish different messages with different retain values.

This **Mqtt** element is new in V2 8.1 firmware, released 4th quarter 2024.

Mqtt.UseTLS

Description: Power PMAC MQTT TLS use control

Units: Boolean

Range: 0... 1

Default: 0

Mqtt.UseTLS specifies whether the Power PMAC's MQTT connection to the broker will use Transport Layer Security (TLS) and Secure Sockets Layer (SSL) or not. If it is set to the default value of 0 when the **`mqttconnect`** function is called, TLS/SSL will not be used.

If **Mqtt.UseTLS** is set to 1 when the **`mqttconnect`** function is called, TLS/SSL will be used for secure connection between client and broker. This requires the proper security certificates.

Power PMAC's selection to use TLS/SSL security functionality or not must match that of the broker.

Server/Client TLS certificates can be created by running the **create_tls_cert.sh** script in the **/opt/ppmac/scripts/mqtt** directory. After certificate creation, the **/opt/ppmac/tools/mqtt/certs** directory will contain the following files:

- **mqtt_ppmac_ca.key**
- **mqtt_ppmac_ca.crt**
- **mqtt_client.key**
- **mqtt_client.crt**

If **create_tls_certs.sh** is run with the **-s** option, the following MQTT broker certificates will also be generated:

- **mqtt_broker.key**
- **mqtt_broker.crt**

In **/opt/ppmac/tools/mqtt/certs**, the **trustlist** folder contains the Certificate Authority (CA) certificates of MQTT brokers that the client trusts. The **mqtt_trusted_ca.crt** file is a concatenation of all the certificates in the **trustlist** folder.

If configuring a broker without pre-generated certificates (e.g. Eclipse Mosquitto MQTT broker), the files **mqtt_ppmac_ca.crt**, **mqtt_broker.key**, and **mqtt_broker.crt** must be copied to the appropriate locations of the MQTT broker.

A TLS/SSL connection requires appropriate certificates and keys to authenticate the broker and the client.

In order to authenticate the broker, the client must have an **mqtt_ppmac_ca.crt** file. In order to provide authentication credentials to the broker, the client must also have an **mqtt_client.crt** file and an **mqtt_client.key** file. These files should be available in the **/opt/ppmac/tools/mqtt/certs** directory.

In order to authenticate the client, the broker must have an **mqtt_ppmac_ca.crt** file. For the Eclipse Mosquitto MQTT broker, this file should be available in the **/etc/mosquitto/ca_certificates** directory. In order to provide authentication credentials to the client, the broker must also have an **mqtt_server.crt** and an **mqtt_server.key** file. These files should be available in the **/etc/mosquitto/certs** directory.

Since the Power PMAC client application uses the same certificate authentication (CA) to sign both the broker and the client, the same **mqtt_ppmac_ca.crt** is used to verify both **mqtt_broker.crt** and **mqtt_client.crt**. Typically, clients will have a list of trusted broker CA certificates and brokers will have a list of trusted client CA certificates.

This **Mqtt** element is new in V2.8.1 firmware, released 4th quarter 2024.

Mqtt.Verbose

Description: MQTT message logging control

Units: Enumeration

Range: 0 .. 4

Default: 0

Mqtt.Verbose specifies which messages associated with MQTT transmissions are logged. The following options are available:

Mqtt.Verbose = 0 Log error messages only
Mqtt.Verbose = 1 Log error messages and events
Mqtt.Verbose = 2 Log all MQTT messages except topic value changes
Mqtt.Verbose = 3 Log all MQTT messages
Mqtt.Verbose = 4 Log all messages including message queue communications

These messages can be found in the **`mqttclient#.log`** file located in **`/var/log`**.

This **Mqtt** element is new in V2 8.1 firmware, released 4th quarter 2024.

MQTT Interface Status Elements

These status elements provide information as to the present state of the MQTT client interface on the Power PMAC.

Mqtt.Connected

Description: Power PMAC connection status on MQTT network

Units: Boolean

Range: 0 .. 1

Mqtt.Connected indicates the connection status of the Power PMAC to a broker on the MQTT network. If it has a value of 0, the Power PMAC is not connected to a broker. If it has a value of 1, it is connected to a broker.

It is set to 1 on the completion of the connection process started by the **`mqttconnect`** function. The user should verify that **Mqtt.Connected** is set to 1 before attempting to publish or subscribe on the network.

This **Mqtt** element is new in V2.8.1 firmware, released 4th quarter 2024.

Mqtt.Error

Description: MQTT communications error status

Units: Boolean

Range: 0 .. 1

Mqtt.Error indicates the error status of the most recent MQTT communication, publish or subscribe.

If it has a value of 0, the communication was successful (no error). If it has a value of 1, the communication was not successful, ending in an error.

This **Mqtt** element is new in V2.8.1 firmware, released 4th quarter 2024.

Mqtt.ErrorID

Description: MQTT communications error number

Units: Enumeration

Range: 0 .. 41

Mqtt.ErrorID indicated the number of the error in the most recent MQTT communications. If it has a value of 0, there was no error in the communications. If it has a value greater than 0, there was an error, and the value specifies which error type occurred.

The following table lists the errors each value specifies. Values of 1 – 31 are standard MQTT errors. Values greater than 31 are PMAC-specific errors.

Error Code/ID	Description
0	No error
1	Out of memory
2	Protocol error
3	Invalid input parameters
4	Client not connected to broker
5	Broker refused connection
6	Callback function not registered for specified event
7	Lost connection to broker
8	TLS error
9	Payload size error
10	MQTT version error
11	User authentication error
12	Access Control List access denied
13	Unknown error
14	System errno error
15	Address information error
16	Proxy error
17	Plugin deferral
18	Malformed UTF-8 string
19	Keepalive timeout error
20	DNS lookup error
21	Malformed MQTT packet
22	Duplicate property
23	TLS handshake error
24	Quality of Service (QoS) not supported
25	Oversize packet
26	Online Certificate Status Protocol (OCSP) error
27	Operation timed out
28	Retain not supported
29	Invalid topic alias
30	Administrative action
31	Item already exists
32	P-variable error
33	POSIX message queue error

34	Set login error
35	Connection timeout error
36	Already connected error
37	Already logged in error
38	Already subscribed error
39	Already unsubscribed error
40	Invalid array length error
41	Other error
42	Reached maximum number of subscriptions

More error information can be found in the **`mqttclient#.log`** file located in **`/var/log`** directory.

This **Mqtt** element is new in V2.8.1 firmware, released 4th quarter 2024.

Mqtt.MsgsRcvd

Description: Number of MQTT messages received

Units: Subscribe messages

Range: Non-negative integer

Mqtt.MsgsRcvd contains the number of MQTT subscribe messages received. It is incremented by 1 each time a new subscribe message is received.

It is automatically set to 0 on the initial connection of Power PMAC to the MQTT broker. It is possible for the user to overwrite the value in the element, for example to reset the value back to 0.

This **Mqtt** element is new in V2.8.1 firmware, released 4th quarter 2024.

Mqtt.MsgsSent

Description: Number of MQTT messages sent

Units: Publish messages

Range: Non-negative integer

Mqtt.MsgsSent contains the number of MQTT publish messages sent. It is incremented by 1 each time a new published message is sent successfully.

It is automatically set to 0 on the initial connection of Power PMAC to the MQTT broker. It is possible for the user to overwrite the value in the element, for example to reset the value back to 0.

This **Mqtt** element is new in V2.8.1 firmware, released 4th quarter 2024.

Mqtt.NumSubs

Description: Number of MQTT topics subscribed to

Units: Subscribe topics

Range: Non-negative integer

Mqtt.NumSubs contains the number of MQTT topics presently subscribed to from the broker. It is incremented by 1 each time a new topic is subscribed to, and decremented by 1 each time a topic is unsubscribed.

It is automatically set to 0 on the initial connection of Power PMAC to the MQTT broker.

This **Mqtt** element is new in V2.8.1 firmware, released 4th quarter 2024.

MQTT Interface Functions

These functions cause actions to be taken by the MQTT client application on the Power PMAC.

mqttclose

Function: Close the Power PMAC connection to an MQTT broker

Syntax: **mqttclose()**

The **mqttclose** function terminates Power PMAC's connection to the MQTT broker, freeing memory that was used with the connection. Status element **Mqtt.Connected** will be set to 0 on a successful disconnection. This function takes no arguments, but the ending parentheses are still required.

The function returns a value of 0 on a successful message request to the broker, or "nan" (not-a-number) on an unsuccessful message request.

This **Mqtt** function is new in V2.8.1 firmware, released 4th quarter 2024.

Example

```
global MqttResult;

MqttResult = mqttconnect("192.168.0.215");

…

MqttResult = mqttclose();
```

mqttconnect

Function: Connect Power PMAC to MQTT broker

Syntax: **mqttconnect(*{string}*)**

where:

- ***{string}*** is the IP address of the broker, or the host name

The **mqttconnect** function attempts to make a network connection with an MQTT broker (server) that has the specified IP address. This function must be called after every Power PMAC power-on or reset before MQTT communications can be used.

The argument is a string that specifies the host broker to be connected to. This can either be the explicit IP address, or the "hostname" of the broker.

A host name can be used as an alias for an IP address by editing the `/etc/hosts` file.

1. Remount the root directory as read-write: `mount -o remount,rw /`
2. Edit the `/etc/hosts` file with vi: `vi /etc/hosts`
3. Add a host name on a new line in the following format:

IP_ADDRESS HOSTNAME

4. Save changes & set the root directory back to readonly: `mount -o remount,ro /`

```
127.0.0.1       localhost
127.0.1.1       ppmac
::1             localhost ip6-localhost ip6-loopback
ff02::1         ip6-allnodes
ff02::2         ip6-allrouters

192.168.5.239   MQTT_Broker
~
```

The **mqttconnect** function returns a value of 0 on a successful connection, or "nan" (not-a-number) on an unsuccessful connection.

This **Mqtt** function is new in V2.8.1 firmware, released 4th quarter 2024.

Examples

```
global MqttResult;

MqttResult = mqttconnect("192.168.0.215");

MqttResult = mqttconnect(MQTT_Broker);
```

mqttpub

Function: Publish data to an MQTT broker

Syntax: **mqttpub(*[{expression}],{variable_name}*)**

where:

- the optional first ***{expression}*** is the number of consecutive global (P) variables whose values will be published. If no number is specified here, the value for only one variable will be published
- ***{variable_name}*** is the name of the first global (P) variable whose value is to be published. It can be the P-variable name itself, or the name of a declared global variable

The **mqttpub** function causes the Power PMAC to publish the value of one or more global (P) variables to the MQTT broker using the topic "global/*variable_name*". This function must be invoked each time a new value of the variable is to be sent to the broker.

The variable name is that of a global (P) variable in the Power PMAC. It can be the name of the variable itself (e.g. **P100** or **P(MqttVars + 5)**), the name of a declared global variable, or the name set by a #define macro.

The function returns a value of 0 on a successful request, or a value of "nan" (not-a-number) on an unsuccessful request.

This **Mqtt** function is new in V2.8.1 firmware, released 4th quarter 2024.

Examples

```
global CycleCount;
global MqttResult;

Mqtt.Prefix = 0;
MqttResult = mqttpub(CycleCount); // The value of CycleCount will be published
                                  // under the global/CycleCount topic
Base=50;
MqttResult = mqttpub(P(Base+20)); // The value of P70 will be published
                                  // under the global/P70 topic

Mqtt.Prefix = 22;
MqttResult = mqttpub(3,P10);      // The values of P10, P11, and P12 will be
                                  // published under the topics 22/global/P10,
                                  // 22/global/P11, and 22/global/P12
```

mqttsetlogin

Function: Set user name and optional password for MQTT connection

Syntax: **mqttsetlogin(*{string}*)**

where:

- ***{string}*** is the user name and (optional) password of Power PMAC as an MQTT client. If a password is provided, it should be separated from the user name with a comma. User names and passwords cannot contain spaces. Login credentials can be reset by providing an empty string ("") for this argument.

The **mqttsetlogin** function specifies the user name, and optionally the password, to be used on a subsequent connection of the Power PMAC to the MQTT broker. If the broker requires one or both of these, this function must be called before the **mqttconnect** function is called.

This function returns a 0 value on a successful message request to the MQTT application, or "nan" (not-a-number) on an unsuccessful request.

This **Mqtt** function is new in V2.8.1 firmware, released 4th quarter 2024.

Examples

```
global MqttResult;
Mqtt.Port = 1883;
MqttResult = mqttsetlogin("admin, 1234");      // Set username "admin"
                                               // Set password "1234"
MqttResult = mqttconnect("MyBroker");          // Establish connection
```

mqttsub

Function: Subscribe to data from an MQTT broker

Syntax: **mqttsub(*[{expression}],{var_name}*)**

where:

- the optional ***{expression}*** is the number of consecutive global (P) variables whose values will be subscribed to. If no number is specified here, the value for only one variable will be subscribed to.

- ***{variable_name}*** is the name of the first global (P) variable whose value is to be subscribed to. It can be the P-variable name itself, or the name of a declared global variable

The **mqttsub** function causes the Power PMAC to subscribe to an MQTT topic "global/*variable_name*" and read the value into the specified Power PMAC global variable. This function only needs to be called once per global variable after the initialization with the **mqttconnect** function. The variable subscribed to will automatically be updated in the Power PMAC when a client publishes to its MQTT topic.

The variable name is that of a global (P) variable in the Power PMAC. It can be the name of the variable itself (e.g. **P100** or **P(MqttVars + 5)**), the name of a declared global variable, or the name set by a **#define** macro.

The function returns a value of 0 on a successful request, or a value of "nan" (not-a-number) on an unsuccessful request.

This **Mqtt** function is new in V2.8.1 firmware, released 4th quarter 2024.

Examples

```
global TargetPos;
global MqttResult;
#define NumOfCycles        P75

MqttResult = mqttsub(TargetPos);
// The value of TargetPos is subscribed to under the global/TargetPos topic

MqttResult = mqttsub(NumOfCycles);
// The value of NumOfCycles is subscribed to under the global/NumOfCycles topic

Base = 75;
MqttResult = mqttsub(1,P(Base+10));
// The value of P85 is subscribed to under the global/P85 topic

MqttResult = mqttsub(2,P500);
// The values of P500 and P501 are subscribed to
```

mqttunsub

Function: Unsubscribe to data from an MQTT broker

Syntax: **mqttunsub(*[{expression}],{var_name}*)**

where:

- the optional ***{expression}*** is the number of consecutive global (P) variables whose values will no longer be subscribed to. If no number is specified here, the value for only one variable will be unsubscribed.

- ***{var_name}*** is the name of the declared global (P) variable that had been subscribed to

The **mqttunsub** function causes the Power PMAC to stop subscribing to an MQTT topic "global/*variable_name*" so it will no longer read the value into the specified Power PMAC global variable. This function only affects topics that have already been subscribed to.

The variable name is that of a global (P) variable in the Power PMAC. It can be the name of the variable itself (e.g. **P100** or **P(MqttVars + 5)**), the name of a declared global variable, or the name set by a **#define** macro.

This **Mqtt** function is new in V2.8.1 firmware, released 4th quarter 2024.

Example

```
global TargetPos;
global MqttResult

MqttResult = mqttsub(TargetPos);

…

MqttResult = mqttunsub(TargetPos);
// The global/TargetPos topic is no longer subscribed to
```

OMRON Corporation Industrial Automation Company

Kyoto, JAPAN Contact : www.ia.omron.com

Regional Headquarters

OMRON EUROPE B.V.
Wegalaan 67-69, 2132 JD Hoofddorp
The Netherlands
Tel: (31) 2356-81-300 Fax: (31) 2356-81-388

OMRON ELECTRONICS LLC
2895 Greenspoint Parkway, Suite 200
Hoffman Estates, IL 60169 U.S.A.
Tel: (1) 847-843-7900 Fax: (1) 847-843-7787

OMRON ASIA PACIFIC PTE. LTD.
438B Alexandra Road, #08-01/02 Alexandra
Technopark, Singapore 119968
Tel: (65) 6835-3011 Fax: (65) 6835-3011

OMRON (CHINA) CO., LTD.
Room 2211, Bank of China Tower,
200 Yin Cheng Zhong Road,
PuDong New Area, Shanghai, 200120, China
Tel: (86) 21-6023-0333 Fax: (86) 21-5037-2388

Authorized Distributor:

Cat. No. O085-E-01 0825