mqTrains USER REFERENCE

CONTENTS

ABOUT USER REFERENCE

This document covers some more technical aspects of mqTrains sketches, some aspects common to all sketches and some more technical information about some of the technologies being used.

ABOUT mqTrains

mqTrains is a collection of programs (called sketches in the Arduino microprocessor world) for controlling model railroad accessories such as turnouts, sensors, lights and signals. These programs can be downloaded in binary form from the web and loaded onto ESP microcontrollers using node-flasher and esptools. Using MQTT as an intermediary server, the accessories can then be accessed by JMRI, Python, JavaScript and potentially other model railroad software. These function independently from train cab control so will function with any DCC system or DC control.

Available now:

- mqTrainsServo - designed for using 9g servo motors for turnouts or semaphores (with an additional 5-6Vdc power supply to drive the servos).

Coming soon:

- mqTrainsIO - primarily designed to connect sensors but can also be used for simple output devices.

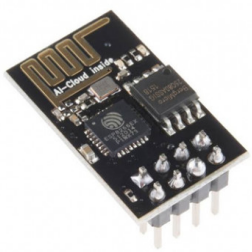
- One-wire signal control using NeoPixels (WS2811 and WS2812b) or multi-wire signal control using standard LEDs.

Visit mqTrains.com for more information.

ABOUT ESP

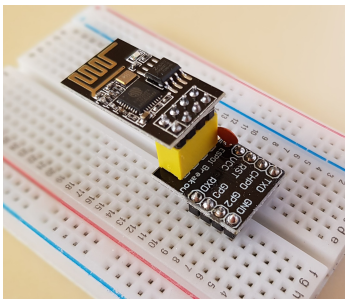
ESP is a term used for a family of inexpensive microprocessor boards based on chips manufactured by Espressif. These are becoming quite common in home automation devices and hobby use. They are quite cheap, small and, with built-in WiFi, they are quite powerful. We have the need for bringing remote control and interaction with parts all across the space of our layout but don't really want the nest of wiring that often comes with it.

They come in a variety of forms. The ESP01 is our favourite because it is the cheapest at about 2 bucks each, packs a lot of punch into about the size of a thumbnail and has a consistent pin layout so is easier to deal with.

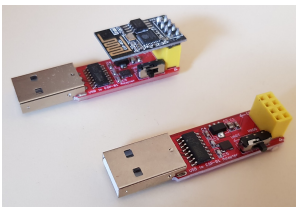


Useful accessories

A breakout board such as the one shown is needed if you want to mount an ESP01 on a breadboard. It may also be used when mounted on your layout so the ESP01 can be removed if need be.



To connect an ESP01 to your computer, you will need a USB Programming Adapter such as these.



And preferred, the one from DYI Mall that does not require sliding a switch to program:

The switch on the side of the programmer, if present, is to toggle between programming mode (marked PROG) and serial monitor mode (marked UART). The switch is only read at the time of power up so the programming procedure is as follows:

1., Move the switch to PROG, unplug the ESP01 from the programmer and plug it back in.

2. Upload the binary as described above.

3. Unplug the ESP01, move the switch back to UART and plug back in. mqTrains will now start running.

ABOUT MQTT

MQTT is a simple messaging system, using an application as a hub, known as a Server or Broker. MQTT clients are apps using MQTT to send or receive messages. In our case, the main MQTT clients are mqTrains on ESP devices and JMRI on Windows, Mac or Linux computers.

All clients send and receive all messages directly to or from the MQTT Server.

MQTT runs over Ethernet and Wireless LAN, so, if you have WiFi, JMRI, the MQTT Server and the ESP01 can communicate with each other over your WiFi instead of wires all over the place. You can place several ESP devices at convenient places around your layout and plug in your accessories. The only wiring you would need to connect these to your JMRI system is 3.3 Vdc power and perhaps another power source to accessories if they need it.

The MQTT Server application can be installed on the same computer as JMRI or another computer of your choice. WiFi-enabled Raspberry Pi computers such as the inexpensive Raspberry Pi Zero W can be used. It is a lightweight app, it's even been called featherweight so it's not a burden on the computer.

We use and recommend using Eclipse Mosquitto as your MQTT Server. This can be installed on Windows, Mac or Linux, including a Raspberry Pi. No special configuration is required for our needs. Refer to https://mosquitto.org/download/ or mqtrains.com for a few short instructions to install Mosquitto on Windows and Linux.

If you prefer a different MQTT Server, feel free to go with that. You can also use your home automation MQTT Server.

---

mqTrains
MQTT in the model train world
A layout with less wires
mqTrains.com

MORE ABOUT ESP

There are two main ESP chip types - the original ESP8266 from 2014 and its successor, the ESP32. There are numerous boards of different shapes and sizes on which these processor chips are mounted.

The ESP01, which uses the ESP8266 chip, is the least expensive and comes in a standard footprint that is not changing with every revision. These are the reasons it is our choice.

We have found that mqTrains also works on NodeMcu boards with ESP8266 chips though they have variants with slightly different pin forms.

The ESP32 processor uses different pins for I^2C so may not work with mqTrains sketches.

If you use a different board or processor, we cannot say or guarantee that mqTrains will work.

An important difference between the ESP01 and the NodeMcu is that the NodeMcu provides DC power to the add-on I^2C modules whereas with the ESP01, it and each of the I^2C modules are powered directly from a 3.3 Vdc power bus.

We only use 4 of the ESP01's header pins. These are:
Ground (black),
I^2C clock (white),
I^2C data (ochre) and
3.3 Vdc (red).

The power source is normally 3.3 Vdc with the maximum permitted being 3.6 Vdc.

ESP devices only use the 2.4 GHz WiFi band. Some common domestic WiFi routers are limited to 16 or 32 concurrent sessions. We are investigating options to share WiFi connections.

MORE ABOUT MQTT

MQTT is a simple, featherweight protocol for sending messages between devices such as small electronic gadgets and computers. It is becoming increasingly common, including in everyday use IOT devices for turning your lamps and power sockets on and off. Messages are sent to the central MQTT Server, commonly called the Broker.

For more background information about the MQTT protocol, this Introduction to MQTT YouTube video covers it fairly well.

MQTT topics and messages

Basics

To send a message to the MQTT Server, in MQTT jargon: a client publishes a message to a specific topic. Receiving a message happens by subscribing to a topic on the Server. Any message that gets published to that topic, is sent to every client that subscribes to it.

MQTT topics are structured as nested levels with the levels separated by "/" characters. There are no rules about how many levels you should use nor what they might mean. That is up to individual circumstances. However, keeping it simple is considered to be best practice.

A message must be published to a specific topic but clients can subscribe to multiple topics by specifying each one or by use of wildcards. The client will then receive messages published to all of those topics. Topics are not client specific so there can be

multiple clients receiving messages from the same topic.

The message for that topic could be an instruction for it or to report something about it. Again, no rules on what messages have to be.

Whilst there are no specific requirements regarding topic names, there are some generally recognised best practices listed below:

- *Never use a leading forward slash in topics*
- *Never use spaces in a topic*
- *Keep the topic short and concise*
- *Only use standard printable characters such as letters and numbers*
- *Use specific topics, not general ones*

[Ref: www.hivemq.com]

So a home automation system might have a topic myHome/Upstairs/Bedroom1/Light/Bedside for the bedside lamp in the upstairs bedroom. A message could be published from a push button sensor, a cell phone app or a voice activated device to tell the light switch to switch on the light.

MQTT retention or persisting

MQTT contains a concept to keep a message in the database even after all the subscribers have received it. This allows your model railroad to "remember" where every turnout was last told to be, what aspect every signal was commanded to, and what the last state of a sensor was. When you publish a message to a topic on the command line with mosquitto_sub, the -r option will do this. The retained, or persisted messages are saved in a file on the computer running the MQTT Server, so even after a Server reboot, the message is still there.

Locking a turnout, for example, is not retained, so only the current subscribers will receive a "lock" command and anyone subscribing later will not know a turnout was locked. This allows you to simply re-power the ESP and the turnout will be unlocked again.

MQTT username and password

For many model railroad cases, an MQTT username and password have been considered not necessary and have not been required. Mosquitto has adopted a higher security level as the default configuration with its release of Version 2.0 in December 2020, requiring logins to access the MQTT Server.

If you have Mosquitto Version 2.0 and prefer to continue without the login requirement, it can be disabled by adding the following to lines to mosquitto.conf in /etc/mosquitto/conf.d/

```
listener 1883
allow_anonymous true
```

If you wish to retain the requirement for login, consult the Mosquitto documentation for details on how to configure it.

mqTrains allows you to set up a username and password on each device. Use the MQTT link from the setup page.

Defining MQTT topics in JMRI

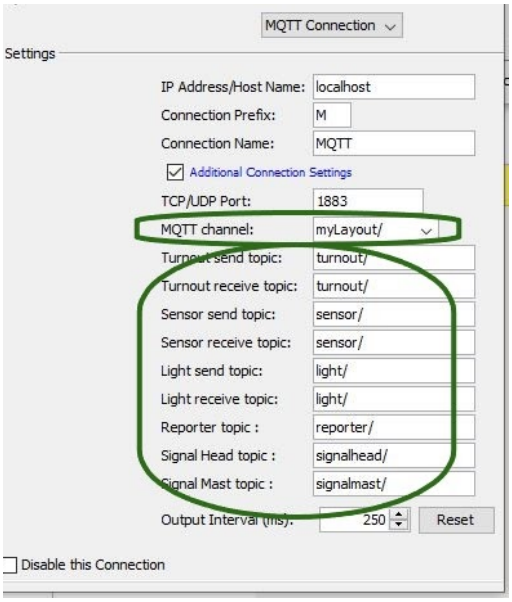
MQTT support for turnouts has been part of JMRI since version 4.12. Sensors and Lights were added in Version 4.21.1. Whilst the setup page also mentions Reporter, Signal Head and Signal Mast topics, these are not yet implemented.

The original topic structure implemented in version 4.12 was quite rigid, it was made more flexible in version 4.15.5 and allowed almost freeform in version 4.21.1. These notes relate to options available in 4.21.1.

MQTT topics in JMRI are defined in three components as shown in the examples below.

The first component is shown as the MQTT channel on the Preferences panel. This becomes a prefix used for all JMRI MQTT elements for this JMRI profile. A typical use of this is a simple identifier for your layout.

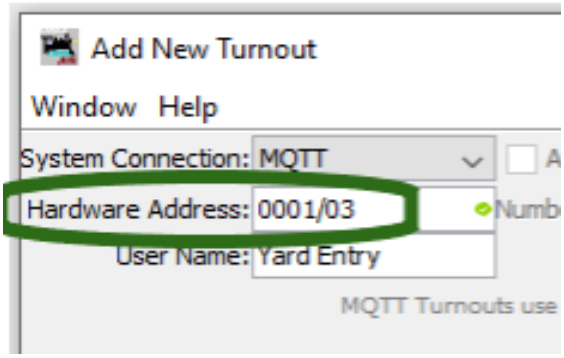
The second component, also on the Preferences panel, is a component used for all MQTT devices of a particular type. For most of these, JMRI allows use of different topics to be used for messages published and messages received from subscriptions.



This allows a more complex conversation to occur between devices but we have found having published and received topics the same is adequate for our purposes. A common use of this component is to use the device type, e.g. turnout or sensor.

For modular system participants with individually owned modules such as Freemo modules, it may be worth considering using a generic or common channel name and standard device type naming in common with others with whom you may combine modules.

The third component is for identifying individual devices and becomes the hardware address for JMRI devices. A typical use of this component is to use an identifier for the microprocessor plus an identifier for individual devices on that microprocessor, as shown in the example below.



Each of the three components can have as many topic levels as you wish, including none or even have one topic level split across device components - they are just concatenated text strings. So you can use whatever naming convention you wish but remember the best practices - don't make long and complex topics.

In the example here, the full topic for this turnout is **myLayout/turnout/0001/03**

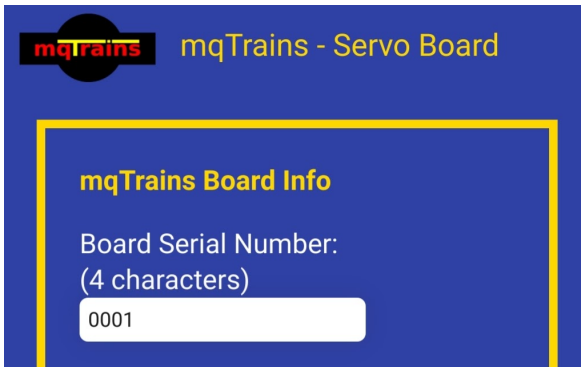
JMRI will create a turnout system name by adding an MT in front of the third component (hardware address). Also remember that all JMRI system names need to be unique.

Defining MQTT topics in mqTrains

MQTT topics in mqTrains are defined in two components. These are the MQTT Base Topic including the Board Serial Number and the pin or sequence number of the individual device attached to the mqTrains board, concatenated in that order as shown below.

The full topic defined in mqTrains and the full topic defined in JMRI must exactly match .

The example below uses the same full topic as in the JMRI example shown above, i.e. **myLayout/turnout/0001/03**



Board Serial Number is shown behind the field for entering the Base Topic (0001 shown). The Serial Number can be changed on the Board Info page or the Quick Start page.

Base Topic is on the MQTT Info page and also on the Quick Start page.

MQTT topic wild cards

MQTT allows two types of wild card. You can use wild cards to receive messages for multiple topics (subscribing) but you cannot use a wild card sending a message (publishing).

"+" means accept any value for one complete level in a topic subscription. It must therefore have "/" separators before and after it unless it is the very first or very last level in a topic.

"#" can only be used at the end of a subscription and means any value for any number of levels thereafter. It must be preceded by a "/" separator and have nothing after it unless used alone.

mqTrains allows you to have one "+" wild card in the base topic. mqTrains appends "/#" after the Base Topic with Serial Number to capture messages for all pins so you cannot add another "#".

SETTING UP YOUR ESP

Setting addresses for add-on modules

I^2C modules such as those used with mqTrains sketches must have unique I^2C bus addresses when connected together. Module manufacturers set a base address on each type of module. A user-modifiable offset gets added to this to make the full I^2C address. The offset number is physically set on the module. This allows more than one of these modules to be attached to master devices such as the ESP01.

The mqTrainsServoPCA sketch uses a PCA9685 module to provide pulse-width modulation (PWM) control of up to 16 servo motors. These modules have a base I^2C address of 0x40. For the offset, there are 6 pairs of solder pads on the module, labeled A0-A5, with each pair being a binary digit - a 0 by default or a 1 by soldering across the gap between the pair. The 6 binary digits provide an address range from 0 to 63 so theoretically, 64 PCA9685 modules could be attached to an ESP01 to drive 1,024 servo motors.

The default offset is zero and as the **mqTrainsServoPCA** sketch only allows one module per ESP, there's no need to change this. If your module has the offset changed, you will need to set that in the mqTrains configuration. **The mqTrainsIO** sketch uses up to 6 MCP23017 modules, each providing 16 I/O pins. The modules are manufactured with a base address of 0x20. The offset is specified using 3 address pins (labelled A0-A2). A pin is set low (connected to GND) for a zero or high (connected to VCC) for a one. The 3 binary digits allow an address offset range of 0 to 7. This offset needs to be set in the mqTrainsn configuration.

Configuring your ESP

Configuring mqTrains is best done using the device's web page. When an ESP device with mqTrains installed first starts, it will start a WiFi Access Point to which you can connect using any computer, laptop, tablet or smartphone. You can then use your device's browser to connect to the web server on the ESP, by going to 2.2.2.1

There are two main components to the configuration. The "System setup" portion handles connectivity parameters to enable WiFi, the Access Point, the MQTT connection and MQTT message parameters. The items most commonly set up are on the **Quick Start** web page. Other pages show more detailed options if required.

The Device Detail configuration portion is used to provide the desired configuration for each device connected.

System setup

This is common to all mqTrains sketches with only minor necessary differences. When altering any of System setup values, type the new value in the relevant box and click on the SAVE button to attempt to make it active. Some System values require the system to reboot, so find the Reboot System on the main page. Table 1 shows the System setup parameters, their default values and a short explanation of them. Figure 1 shows a snapshot example of these parameters on the web page.

When the mqTrains system successfully connects to the MQTT Server, it will post it's IP address to the ../IPA topic like in this example:
TxNamib/servo/0011/IPA 10.10.10.148 is here!

If you are monitoring the MQTT Server with an app or with mosquitto_sub, you will see this message being published.

If you need to change some configuration on the ESP, you will need to connect your computer, tablet or smartphone back to your WiFi router and use the browser to the IP address shown.

Configuring can also be done over a serial cable connected between a computer and the ESP board. Some configuration can also be done using MQTT commands.

System Parameters

This is common to all mqTrains boards:

Parameter	Default	Comments
mqTrains Board Info		
Board Serial Number (NNNN)	0001	Serial numbers should be unique and will become part of the MQTT Topic. Set this to a unique number of your choice. Make sure the Topic is defined correctly in JMRI to include this number.
Software Version		This will show which mqTrains sketch is installed and its version number.
MAC address	-	This is assigned by the ESP manufacturer and is not changeable.
MQTT Info		
MQTT Server IP address	10.10.10.13	The IP address of the device on which the MQTT Server is installed.
MQTT IP port	1883	The IP port number defined in the MQTT Server set up. 1883 is the default.
MQTT Username	-	MQTT server login details. These can be omitted unless your MQTT Server has been configured to require account login. The most common set up for our purposes is not to have account logins. *(The default in mosquitto version 2.0 requires authentication, but until JMRI supports authentication too, please disable this in the mosquitto.conf file.)*
MQTT Password	-	
MQTT Base client name	mqTrains	MQTT clients require a unique client name when connecting. The ESP will use the base client name specified and append the last two portions of the IP address. The default should be adequate but you can change it if you wish.
MQTT Base Topic	myLayout/ *type*/ NNNN added for Serial Number	The base topic is the first portion of the MQTT topic. It can contain multiple levels. The board's serial number will be appended to this and called the "Base Topic". The ESP will subscribe to the Base Topic plus "/#". Adding the I/O identifier creates the full MQTT topic to which the ESP will subscribe. The full topic defined in JMRI must match this subscription. A '+' wild card is permitted for **one** level in the topic. This allows you to use different topics for different device types in JMRI. Example myLayout/**light**/0001/**01** and myLayout/**turnout**/0001/**02** will control two adjacent pins on the same 0001serialized ESP if the Base Topic is set to myLayout/+/0001/**01** Messages processed will be echoed back to MQTT using the topic 'mqtt_callback'.
MQTT Publish Topic		This is for display only, derived by concatenating the Base Topic and the Board Serial number defined above. If a '+' wild card has been used, this will be replaced with *type*. This topic will be used by mqTrains when publishing updates.
WiFi Info		
WiFi SSID	Linksys	The SSID and Key of the WiFi service to use for ESP devices to connect to the MQTT Server.

WiFi Key	1234123412 (never shown)	The Key will only be displayed while entering a new one (so that you can check for typos). For your security, the key will otherwise not be displayed. The Key must be 8 or more characters.
Access Point Info		
AP SSID	mqTrains-nnnn	The name of the ESP device's WiFi Access Point. Each device should use a unique SSID so you know which one you are connecting to. The Board Serial Number is automatically appended to the value you enter here.
AP Key	mqtrains	The default Key for the Access Point. To be more secure, you should change this to something of your choice. The Key must be 8 or more characters.
Access Point Enabled	False (Disabled)	The Access Point will turn on during every power up, and remain on until there is a message successfully published to the MQTT Server. If Access Point Enabled is set here, it will remain on, otherwise it will shut down at this point. The Access Point is always turned on during reboots so that it can be used if need be to access the web server to change WiFi details if your WiFi has changed and can no longer be reached.
AP IP address	2.2.2.1	This is defined within mqTrains and cannot be changed within the menus. Put this address into a browser to call up the configuration web page.
Note: *type* is sensor on IO modules and turnout on Servo modules.		

Table 1 - System setup parameters

ServoPCA Module set up

Having established the WiFi and MQTT connections, the next job is to configure the servo modules and pins.

Table 2 shows the servo module setup parameters, their default values and a short explanation of them. Figure 2 shows a snapshot example of these parameters on the web page.

When altering any module parameters, untick online first type, update settings as required, tick online to activate.

Press SAVE to persist the active settings to the ESP01's internal storage, then press Reboot System in the main menu to restart the ESP and refresh your web page.

ServoPCA Module operation

mqTrainsServoPCA responds to MQTT messages matching the subscribed topic specified. The last portion of the topic in the message received is used as the servo number and the payload in the message indicates whether a turnout should be thrown or closed.

For example, the message:
myLayout/turnout/0001/04 "THROWN"
will instruct the servo on the 4th PWM terminal to move the turnout to its thrown position.

There are also some special commands that can be sent to servo topics from ../01 to ../16 as shown in the Special MQTT messages section below.

Servo module parameters

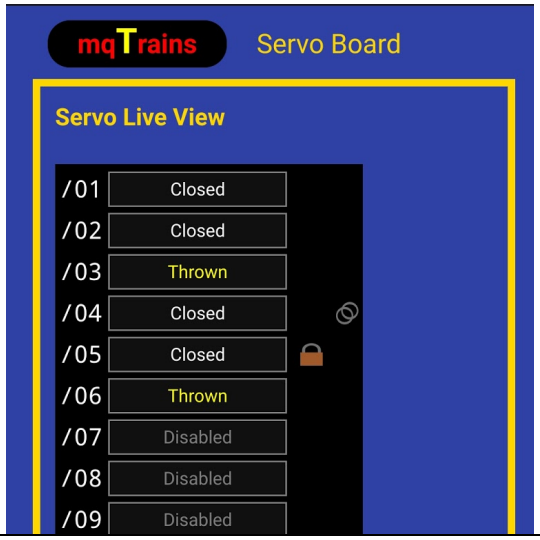
Parameter	Default / Valid Range	Comments
Servo Module Setup (PCA9685)		
Supported modules	1	mqTrains PCA9685 module currently supports 1 module.
PCA module	1	The module being configured below (only one support now)
I²C base	0x40	This base I²C address is set by the manufacturer and cannot be changed.
I²C Address Offset	0	I²C addresses for PCA modules have a base hexadecimal address of 0x40. Each module has solder pads that allow different address offsets so multiple modules can be used. We currently use only one module so this is the default offset. If your module already has a different offset set, put that offset here.
Max number of servos	16	All 16 PWM outputs are supported as servos
Update time	30ms	This controls the speed of the servo motor, the bigger the number, the slower it moves. The number is how often each servo is sent a new value to move closer to the commanded state. If a servo needs to move from position 200 to 300, and the update value is 100, it will take 10 seconds to complete the move from one end to the other.

Servo Detail		
Servo Number	/nn (01-16)	Servo numbers are assigned sequentially, starting with /01. This is the final component of the MQTT topic. Please note that mqTrains counts from 1 whereas the PWM terminals on the module are labeled counting from 0 so servo /01 connects to PWM terminal 0 on the PCA module.
Closed Position	200 (115-500)	The Closed Position value is specifying where the servo needs to move to Close the points in the turnout. This is also called Normal or Straight and the output can also be moved here with an Inactive message. The Thrown Position value is specifying where the servo needs to move to Throw the points in the turnout. This is also called Reversed, Open or Diverging and the output can also be moved here with an Active message. The 50 Hz PWM pulstrain on each output is controlled by a 12-bit value. But a servo pulse needs to be between 1 ms and 2.5 ms long to move the servo from -90 degrees to +90 degrees. The minimum and maximum value of this field is set to 115 and 500.
Thrown Position	240 (115-500)	
Enabled	Yes (Yes/No)	Check the box to enable the servo. Disabled servos won't respond to any request.
Locked	No (Yes/No)	This feature will prevent a turnout from changing. It is intended for operating sessions where the dispatcher might need sole control of main line turnouts. It provides extra protection from users inadvertently switching turnouts. The locked state can not be saved to the configuration flash, and should not be "persisted" in MQTT, so on power up, everything is unlocked. This prevents requiring a dispatcher to unlock previously locked turnouts. A simple power cycle will unlock all turnouts.
Linq to	0 (0,1-8)	This allows two or more servos to be controlled together, typically for a crossover. Disabled when set to 0. If set to 1 through 8, a message to topic ../L1 through ../L8 would update all the servos Linqed to it. For example, if servo /01 is Linqed to L2, then a CLOSED message to L2 would set /01 to CLOSED.A servo can only be assigned to 1 linq.
Invert Linq	No (Yes/No)	If set, mqTrains will invert the request for this linqed turnout, i.e. As in the example above, if the linq receives a CLOSED request, this inverted linqed turnout will move to the THROWN position.

Servo Live View

This shows name and state for each servo and displays icons for locked and linqed servos.

Tapping on the state of an enabled servo will flip it and also publish an MQTT message so that JMRI will be notified of the change.



IO Module set up

Each pin can be configured for use as input for a sensor or as output to control a turnout, light or other device.

Table 2 shows the IO module setup parameters, their default values with the range of valid values and a short explanation of them.

Parameter	Default	Comments
I/O Module Setup (MCP23017)		
# of IO modules	1 (1-6)	Specify the number of MCP23017 modules you have connected. Up to 6 modules are allowed.
i2c addr	0 (0-7)	Adjust the I²C address if need be. MCP23017s have a base address 0x20 to which the number here is added for the real i2c address. Example: 1 here would use 0x21 in the system.
I/O	-	Pin numbers are assigned sequentially, starting with /01. This is the final component of the MQTT topic. The numbers continue increasing on subsequent MCP23017 modules. Up to 6 modules are supported, allowing numbers upto /96. Note that whilst each module has pins labeled A0-A7 and B0-B7, mqTrains counts from 1 so pin /01 is labeled A0 on the MCP23017 module.
State	-	This shows the current state of the pins. 0=Low (Inactive or Off), 1=High (Active or On). These are dynamically updated every few seconds so you can watch what's happening.
Enable	Enabled	Disable any pins not in use. This will block any MQTT messages from being processed for those pins.
Mode	Input (or Output)	Set each pin as input or output for your requirements.
Locked	Unlocked	Not used on IO modules yet
Offline/ Online	online	Uncheck the online/offline check box to take the modules offline to avoid conflict while adjusting module and pin settings,
Update Module		Press Update to activate the changes you made.
Remove Module		Remove this module from your system. Only the last module can be removed. If there is only one module, it cannot be removed.

Table 2. IO Module setup parameters

Special MQTT messages:

Special servo messages:

`../nn "Cxxx"` Sets the Closed position for turnout nn to xxx* and moves to that position, even when Locked. The new value has not been persisted yet, see `../00 "SAV!"`

`../nn "C+10` Sets the Closed position to 10 more* than its current value and moves to that position. The new value has not been persisted yet, see `../00 "SAV!"`

`../nn "C-5"` Sets the Closed position to 5 less* than its current value (if allowed) and moves to that position. The new value has not been persisted yet, see `../00 "SAV!"`

`../nn "Txxx"` Sets the Thrown position for turnout nn to xxx* and moves to that position, even when Locked. The new value has not been persisted yet, see `../00 "SAV!"`

`../nn "T+10` Sets the Thrown position to 10 more* than its current value and moves to that position. The new value has not been persisted yet, see `../00 "SAV!"`

`../nn "T-5"` Sets the Thrown position to 5 less* than its current value and moves to that position. The new value has not been persisted yet, see `../00 "SAV!"`

`../nn "L"` Lock the Turnout

`../nn "U"` Unlock the Turnout

** position commands will only work if the new value is in the allowed range (115-500)*

Special ESP messages:

`../00 "SAV!"` Save the servo positions and the system configuration to the ESP flash memory

`../00 "RPT?"` Request the current states of all turnouts to be published again

`../00 "PCFG?"` Report the JSON configuration of the current setup

`../00 "IPA?"` Report the IP Address of the ESP

`../00 "ENA?"` Report, 16 bit hexadecimal value, all enabled servos. Default: 0xFFFF all enabled.

`../00 "LCK?"` Report, 16 bit hexadecimal value, all locked servos . Default: 0x0000 all unlocked.

`../00 "VER?"` Report the software version and board type of the mqTrains code

`../00 "RST?"` Reboot the ESP

Also note that every special command to ../00 will have the '?' and '!' marks replaced with an '_' and republished by the ESP. This prevents the ESP from executing the command twice and shows the original sender that the command was executed.

Backup and Restore Config

mqTrains configuration details are stored in JSON formatted files on the ESP device.

- localsysconfig.json contains the ESP board (and PCA module) settings
- servoconfig.json contains the servo settings
- And less important sysconfig.json contains the factory defaults

Backup copies of these files can be made by either of these two methods.

1) View via the Web Server page using the IP address followed by the JSON file name and then copy and paste to a file on your computer.
2) Using the curl line command.

Examples:

1) Paste 192.168.20.107/servoconfig.json or 192.168.20.107/localsysconfig.json in a browser and save the json text to a file.

2) `curl 192.168.20.107/servoconfig.json` and save the output to a file.

To save ESP with serial number 0002 use:
`192.168.20.107/localsysconfig.json > 2021.01.31_0002_pca_localsysconfig.json`

And now you have the date, the serial number, the type of board and the specific json file captured.

A process for uploading a json configuration file is being developed.

Over the Air (OTA) upgrades

A process for upgrading the mqTrains firmware "over the air", i.e. to use WiFi instead of a USB connection, is being tested.