Understanding the V4.09 Firmware Upgrade for 2500P-ACP1

Overview

The February 2019 release of V4.09 firmware for 2500P-ACP1 marked one of the most significant upgrades in product features since V3.03 introduced support for Ethernet/IP communications. This Tech Tip examines this upgrade in more detail.

New Features in V4.09

Two significant new features were added in V4.09:

1. IP Aliasing, which allows the module to connect to two different Ethernet networks simultaneously. This feature is very useful when the ACP1 is communicating on a device network (for example, a network of Ethernet/IP drives) and you want to keep this network isolated from the PLC. Complete details on this feature are included later in this document.

2. MQTT Client protocol, which brings the capability for 2500 Series® Processors to communicate on IIOT networks. A summary of this feature is included later in this document, and an upcoming Application Note will show details about how to use this protocol to communicate with an MQTT broker.

In addition to these new features, a number of enhancements were made, covering many areas in the product:

- Added message for notification that module reset was required when making changes to IP Address and/or Subnet Mask fields in Module Configuration web page

- Increased number of variables managed by the Network Data Exchange task to 2000 and increased the publish rate to 210 data points per program cycle.

- Added a new error (Error 392) for detection of an Invalid Array Index accessed in user program. Included in ACP1_STATUS function and web server Error Descriptions page.

- Set 'Incompatible Application' (Error 175) status to create event log message, display on module front panel, show in Error Descriptions page, and in CTI Workbench 'Runtime' window if application program downloaded to ACP1

## ROCK SOLID PERFORMANCE. TIMELESS COMPATIBILITY.

includes function blocks not supported by runtime.

- Increased runtime system memory limits to support 1.5MB of code and 2MB of database. This expansion helps users of large Ethernet/IP networks who were running short of memory.

- Updated runtime system to Straton V9.2 adding following features: Support for Dynamic Memory Allocation and Management of Text Buffers, Functions to read/write arrays to file, Functions supporting Ethernet/IP Explicit Message command.

- Resolved a problem where the Event Log could become corrupted when simultaneously logging events from different code sections.

- Resolved a problem where CTI Workbench connection attempts were rejected due to incorrect count of active server connections .

- Resolved a problem where module resets when using 'Clock' function blocks in user program without setting RTC.

- Resolved a problem where module resets when CAMP Client detects a TCP socket error .

- Resoved a problem where module resets when FTP Server is unreachable when file transfer triggered (Error 412) .

- Resolved a problem where module resets when EIP Tag Client encounters communication loss with server .

- Resolved a problem where module rejects TCP connections caused when TCP sockets were not released when EIP Tag Client connections were closed.

- Changed ingress and egress data rate limit settings in the on-board Ethernet switch.

- Disabled 'Fast Aging' feature to prevent Ethernet unicast packets from being forwarded to all ports.

- Change to I/O configuration now takes effect only after module reset.

- Removed memory inefficiencies in I/O configuration.

- Resolved a problem where module does not work (using ACP1 I/O interface) in a Profibus network with 2500-RBC base controller.

- Resolved a problem in the DataCache interface where some values not written when transfer buffers limit exceeded

- Resolved a problem where the Modbus Client writes zero values on first transaction after program start.

- Resolved a problem where Modbus Client UDP communications timeout is not detected.

- Resolved a problem where Modbus Client error status is not cleared if Modbus Client was not in project configuration.

- Resolved a problem where IP Address validation generated an error in Module Configuration web page.

Using IP Aliasing to Connect to Two Different Ethernet Networks

The new IP aliasing feature allows assignment of two IP addresses to the ACP1. When combined with the already existing port isolation feature, this new capability allows the ACP1 to be simultaneously connected to two separate Ethernet networks, as shown in the diagam below. This can he helpful in situations where you want to maintain isolations between your device network (Ethernet/IP drives, for example) and the PLC network.
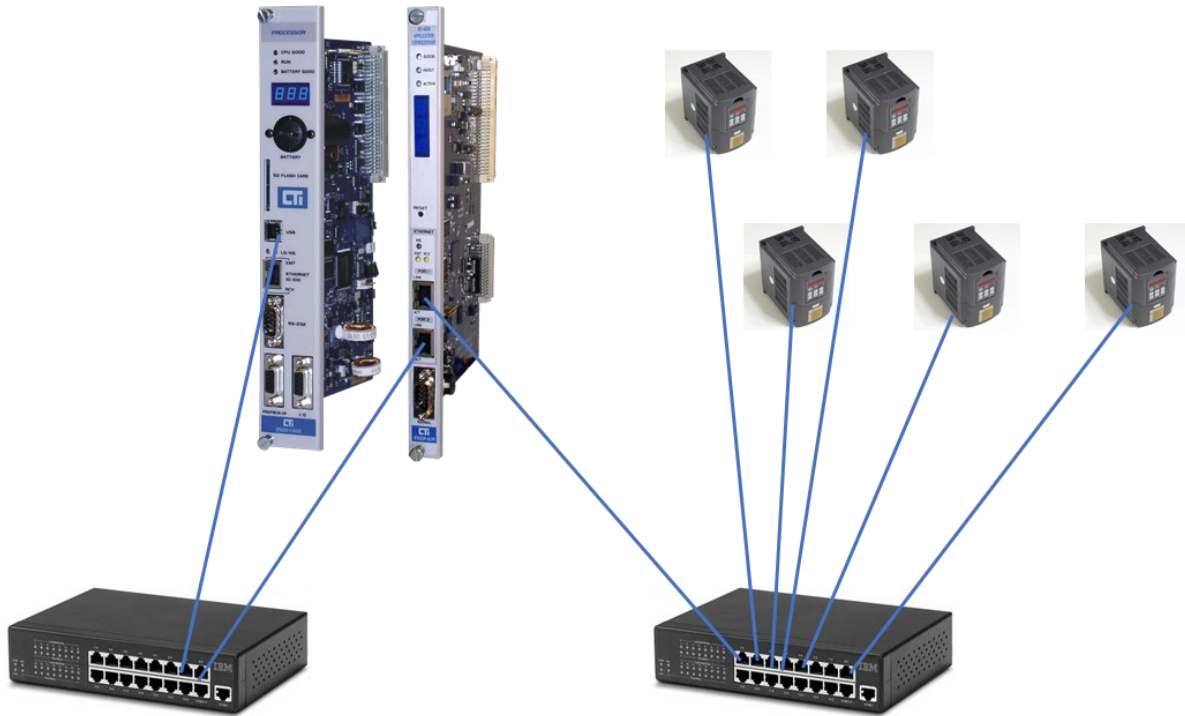
IP aliasing is easy to enable, requiring just two steps.

1. Place SW4 in the CLOSED position to enable port isolation.

2. In the module web page, browse to the "Module Configuration" page. Enter the desired parameters for IP address and subnet mask on the second network. Click Submit. A module reset will be required for the new configuration to take effect.

When using port isolation with IP aliasing, it is not important which network is plugged into which physical ports on the module.

Using MQTT Client Protocol to Communicate on IIOT Networks

The Internet of Things is a popular topic at industrial conference where network communications are discussed. Increasingly affordable micro controllers like Arduino and Raspberry Pi are enabling cheap devices that measure sensor data and send it over the networks.

The term Internet of Things was first used by Kevin Ashton in 2009 for interconnecting physical devices over the internet. The basic idea is very simple: Physical devices can exchange data between each other or be controlled by others. Examples of such devices would be a refrigerator, a car, a building or basically any other electronic device. One of the most common use cases is the collection, transmission, consolidation and displaying of sensor data. The results could be a web dashboard with the aggregated values or an alarm, when a threshold is exceeded. The application scenarios are almost unlimited. Imagine your alarm clock would know that your train to work is 15 minutes late and adjust itself accordingly. Also your coffee maker is switched on automatically 15 minutes later to make you a hot cup of coffee before you leave for work. Sounds like the future ? All that is already possible today. Ericsson predicts that in 2020 50 billion devices are connected over the internet. The communication between the huge amount of devices is enabled by IPv6 and lightweight communication protocols like MQTT.

MQTT was developed by Andy Stanford-Clark (IBM) and Arlen Nipper (Eurotech; now Cirrus Link) in 1999 for the monitoring of an oil pipeline through the desert. The goals were to have a protocol, which is bandwidth-efficient and uses little battery power, because the devices were connected via satellite link and this was extremely expensive at that time. The protocol uses a publish/subscribe architecture rather than typical request/response method. Publish/Subscribe is event-driven and enables messages to be pushed to clients. The central communication point is the MQTT broker, it is in charge of dispatching all messages between the senders and the rightful receivers. Each client that publishes a message to the broker, includes a topic into the message. The topic is the routing information for the broker. Each client that wants to receive messages subscribes to a certain topic and the broker delivers all messages with the matching topic to that client. Therefore the clients don't have to know each other, they only communicate over the topic. This architecture enables highly scalable solutions without dependencies between the data producers and the data
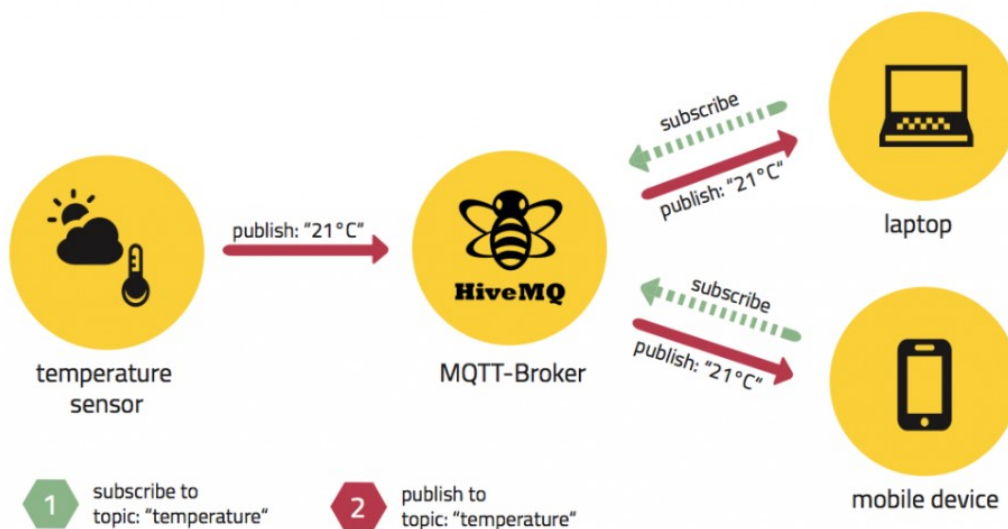
consumers.

MQTT fits well with the CTI data-sharing architecture since publish/subscribe is the basis for our Network Data Exchange protocol.

Unlike other protocols, a client doesn't have to pull the information it needs, but the broker pushes the information to the client, in the case there is something new. Therefore each MQTT client has a permanently open TCP connection to the broker. If this connection is interrupted by any circumstances, the MQTT broker can buffer all messages and send them to the client when it is back online. As mentioned before the central concept in MQTT to dispatch messages are topics. 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.

Fundamentally, that's all there is to MQTT. The only thing needed to set up communications based on MQTT is an MQTT broker and devices which support MQTT. Free broker software can be downloaded from numerous sources. CTI uses Mosquitto. There is also commercial software available for those wanting a solution with support.

Source: MQTT 101 - How to Get Started with the lightweight IoT Protocol. Written by The HiveMQ Team, October 22, 2014

CONTROL TECHNOLOGY, INC.

5734 Middlebrook Pike
Knoxville, TN 37921 USA
+1.865.584.0440
www.controltechnology.com
sales@controltechnology.com

22FEB2019

ROCK SOLID PERFORMANCE. TIMELESS COMPATIBILITY.