MECCO® CONNECT MARK TRACE®

ethermark®

MODBUS-001

COMMUNICATION GUIDE

www.mecco.com

Purpose

This document provides guidance on using MODBUS protocol with EtherMark™, to control MeccoMark® laser marking systems, and Couth dot peen marking systems.

Scope

This document explains how to use MODBUS protocol to control an EtherMark system in a production environment. MODBUS and PLC programming is beyond the scope of this manual. An operator should obtain specific training on PLC logic before attempting to program a system in a production line.

Introduction

EtherMark is a network protocol developed by MECCO Marking and Traceability, specifically to enable MeccoMark® and Couth marking systems to be controlled by programmable logic controllers (PLCs) using industry standard APIs (application programming interfaces). Using EtherMark enables your product marking systems to be integrated into a factory's existing EtherNet/IP, Profinet, or Modbus automation network. Marking jobs can be programmed using standard PLC logic, eliminating the need for operators to learn a new programming language.

Modbus is a communications protocol originally developed by Modicon, for use with PLCs and industrial automation systems. Modbus is currently promoted by the Modbus Organization (www.modbus.org), a group of independent automation systems suppliers and users.

Reference Documents

The reader is directed to supplemental information found in the following reference documents:

EtherMark Control Protocol User Manual

EtherMark Object Model Modbus Register Map

Overview

Modbus is a master/slave application layer messaging protocol, originally designed for serial communication networks. A Modbus master device, such as a PLC, initiates commands (requests) to one or more Modbus slave devices. Upon receiving a request, a Modbus slave executes the request, and sends a reply back to the Modbus master. All communication transactions are initiated by the Modbus master device.

Modbus communication transactions are composed of PDUs, or Protocol Data Units. A PDU combines a one-byte function code and a variable length data field, with a maximum payload length of 253 bytes. The Modbus standard organizes data in 16-bit registers, transmitted in big-endian format (high-order byte first).

With the advent of Ethernet-based TCP/IP networks, Modbus/TCP protocol was developed, to permit Modbus protocol to travel across a standard TCP channel on port 502. Modbus/TCP simply

encapsulates the standard Modbus PDU in an IP packet. The Modbus address and checksum in the PDU are ignored, and are superceded by the IP address and packet checksum.

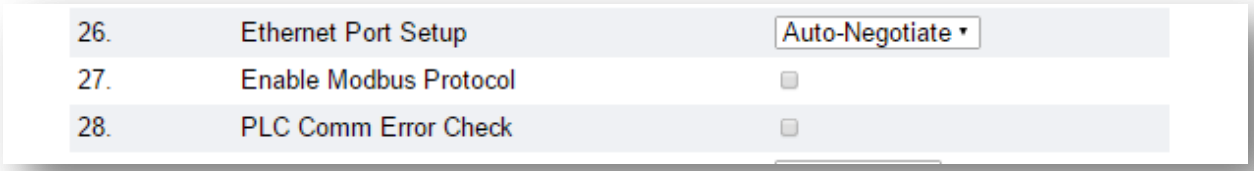
When using Modbus/TCP, the nomenclature convention is as follows: The Modbus master initiating commands is termed the Modbus client. The Modbus slave responding to commands is termed the Modbus server.

Configure EtherMark for Modbus

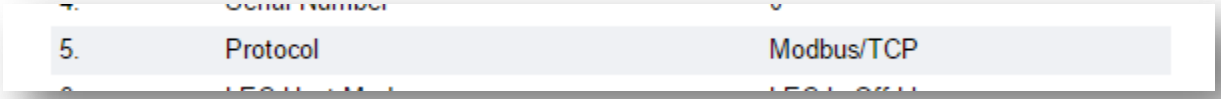
There are two methods to enable the Modbus protocol. Both require the user to know the EtherMark card's IP address. The two methods are described in the Web Utility and Telnet sections below.

Web Utility

Navigate to the IP address of the EtherMark in any web browser. This will bring you to the main screen of the web utility. From here you will select **Configuration** to access the settings page. Line 27 under **Setpoints** will have a checkbox for "Enable Modbus Protocol."



To enable Modbus, select this checkbox and hit **Save** at the bottom of the page. After EtherMark is rebooted, navigate back to **Configuration** and ensure that the **Protocol** on Line 5 has changed to Modbus/TCP.



The EtherMark system can now be fully controlled via Modbus protocol.

Telnet

Login to the EtherMark with a telnet client program (such as putty). Press ENTER a few times, and then give the following commands to enable Modbus protocol, and then reboot the EtherMark:

```
modbus on
reboot
```

Close the telnet client program. The EtherMark system can now be fully controlled via Modbus protocol.

EtherMark Modbus Implementation

The EtherMark system conforms to the Modbus big-endian standard, where 16-bit registers are transmitted high-order byte first. Details and techniques for controlling the EtherMark via Modbus are described in the sections that follow.

EtherMark Object Model, Modbus Holding Registers

The EtherMark system exposes an object model containing the following two data structures:

- O2T Output Block - Content written by Modbus client, read by EtherMark
- T2O Input Block - Content written by EtherMark, read by Modbus client

The Modbus client performs supervisory control of the EtherMark system, by writing control data to the O2T output block, and reading status data from the T2O input block. Please refer to the *EtherMark Control Protocol User Manual* and *EtherMark Object Model Modbus Register Map* documents for detailed descriptions of these data structures.

The EtherMark system provides access to the O2T and T2O blocks using the Modbus Holding Register technique. The O2T output block is mapped to Holding Register address 01000, and contains 218 16-bit registers. The T2O input block is mapped to Holding Register address 02000, and contains 196 16-bit registers. The *EtherMark Object Model Modbus Register Map* documents the O2T and T2O data structures, and provides the Modbus Holding Register address for each element stored within the data structures.

Provided the EtherMark system is configured with the *modbus on* command, the Modbus client can employ Modbus function code 06, Write Single Register, and function code 16, Write Multiple Registers, to write content to the O2T output block.

The Modbus client can employ Modbus function code 03, Read Holding Registers, to read content from the T2O input block. The Modbus client can read back the contents of the O2T output block as well. The Modbus client can always read data from the EtherMark system, *whether or not* the EtherMark is configured with the *modbus on* command.

Modbus Holding Register Address Convention

Modbus convention defines Holding Register addresses, from the Modbus client's perspective, to be in the range 40001 to 49999. Internal to the EtherMark, and encoded in Modbus PDU packets, the O2T output block is at address 01000, and the T2O input block is at address 02000.

To convert internal EtherMark addresses to Holding Register addresses employed in the Modbus client, simply add 40001 to the address. Thus, the Modbus client should specify Holding Register addresses as follows:

- O2T Output Block Holding Register Address 41001
- T2O Input Block Holding Register Address 42001

These addresses are given in the leftmost column titled *Modbus Convention*, in the *EtherMark Object Model Modbus Register Map* document.

EtherMark Object Model Data Types

The O2T and T2O blocks are compound data structures, that contain many smaller nested data structures and simple data types. Please refer to the *EtherMark Object Model Modbus Register Map* document for details.

The most common data types are the *int16* signed 16-bit integer, and the *uint16* unsigned 16-bit integer, which each fit cleanly into one Modbus holding register. The *uint32* unsigned 32-bit integer data type is stored in two adjacent Modbus holding registers, with the high-order bytes at address N, and the low-order bytes at address N+1. For data type details, please see Table 3 in the *EtherMark Object Model Modbus Register Map* document.

The CIP_STR32, CIP_STR64, and CIP_STR128 data types contain ASCII text strings. The first two holding registers contain a uint32 (32-bit integer) str_len field, which records the length of the string. The remaining holding registers contain the string itself. Strings are packed two 8-bit characters to each 16-bit holding register. The first character is stored in the high-order byte of the holding register. It is recommended that strings be null-terminated (last byte value is 0).

The CIP_INDEX_UNIQUE data type contains four discrete byte variables, stored in two holding registers. These variables control the following features: serial number uniqueness (duplicate detection), Cognex camera barcode and OCR verification, and text justification.

For string and nested structure details, please see Table 4 in the *EtherMark Object Model Modbus Register Map* document.

Modbus/TCP Client PC Software

To facilitate testing the EtherMark with a Windows PC, a variety of Modbus/TCP client software is available. The following two software packages were tested to work with EtherMark. The first tool is free, and the second has a 30 day free trial period:

- Ananas Modbus/TCP Client: http://www.tuomio.fi/ananas/
- ModbusTools Modbus Poll Client: http://www.modbustools.com/modbus_poll.html

Example: Read EtherMark Heart Beat

Install Modbus/TCP client software on your PC, or engage Modbus/TCP protocol in your PLC. Start up the Modbus/TCP client. Configure it with the EtherMark system's IP address, and then establish a Modbus connection to the EtherMark.

Please refer to Table 2 in the *EtherMark Object Model Modbus Register Map* document. The data structure described in this table conveys EtherMark status data. The Modbus holding register addresses appear on the left hand side.

Configure your Modbus client to read the 16-bit Holding Register at internal address 02009 (or Modbus convention 42010). This register represents the low-order 16-bits of the EtherMark's 32-bit heart_beat variable. Each time this register is read, it will return a new value, confirming that the EtherMark is alive and well.

Here are the steps for using the Ananas Modbus/TCP client software to perform this test:

1. Start the Modbus/TCP client software: execute **Ananas.exe** (or **Ananas64.exe**)
2. Locate the **Client** box at the top right, and click the **Show** button inside the box.
3. In the **Modbus/TCP Client** window that appears, click the **Connect…** button.
4. In the **Server connection** window that appears, enter the EtherMark's IP address.
5. The **Connection partner** box should show the EtherMark's IP address, port 502.
6. In the **Read Input/Holding register** box, click the **Holding registers** radio button.
7. In the input box below the radio button, enter the address **2009** and click the **Add** button.
8. In the bigger Ananas window, locate the **Start address** input box at top center.
9. In the **Start address** input box, enter the address **2000**.
10. Locate the value of the register at address **2009** in the **Registers** table on the left.
11. Observe that the Heart Beat value at address 2009 increments continuously,

Observing the Heart Beat register incrementing continuously, confirms that the EtherMark system is alive, and that the Modbus/TCP protocol link is working.