AN133

16-124795rev00

Modbus over TCP/IP

Introduction

Version 4.18 firmware adds support for sending Modbus commands over TCP/IP on the Copley AEV drive. This feature will be supported in future EtherCAT plus drives based on the AEV design.

Enabling Modbus mode

Modbus over TCP/IP is available in the drive when the drive is set to standard Ethernet mode. Standard Ethernet mode disables EtherCAT support and enables a number of different Ethernet protocols such as UDP command, DHCP support, and Modbus over TCP/IP. Applications note AN121 contains additional information about the other standard Ethernet protocols supported by Copley drives.

Standard Ethernet mode is enabled when bit 9 of drive parameter 0x121 (Network Configuration) is set. Setting this bit disables standard EtherCAT network handling and enables Modbus and other standard Ethernet protocols.

Modbus overview

When the drive is enabled in standard Ethernet mode, it listens on TCP/IP port 502 for Modbus communications. The drive supports a single Modbus connection, so only one client may connect to the drive's Modbus server at any time.

Modbus communications consist of either single bit accesses (called coils or discrete inputs), or 16-bit register accesses. Copley's implementation does not support any single bit accesses, all Modbus communications consist of reading and writing 16-bit registers.

Modbus further defines two different types of 16-bit registers. Input registers are read only, holding registers are read/write. Copley's implementation treats input registers and holding registers identically. Reading input register 0 is exactly the same as reading holding register 0, both access the same internal drive parameters.

Copley's Modbus implementation allows any drive parameter to be read or written. Additionally, several other drive functions can be accessed by reading or writing specific 16-bit Modbus registers.

Unit identifier

Modbus over TCP/IP includes a byte called the unit identifier in the command header. This identifier is used to select which device the Modbus command is directed to when sent through a gateway device.

Copley uses this identifier to select which axis of a multi-axis drive is being addressed. For single axis drives this value must be set to zero. Any other value will result in an error.

Register mapping

The Modbus register space consists of 65536 registers numbered from 0 to 65535. Copley's Modbus implementation maps different sections of this register space to different drive functions as follows:

Start	End	Function
0x0000	0x03FF	Standard Copley parameters mapped as 2 registers each.
0x0800	0x0BFF	Standard parameters from flash memory.
0x1000	0x107F	Group of registers used to access parameters longer then 32-bits.
0x2000	0x200F	Special registers used to perform commands.
0x3000	0x303F	Copley Indexer registers

Parameter access

Modbus registers in the 0x0000 to 0x03FF range map directly to Copley parameters as described in the Parameter Dictionary document.

Two 16-bit Modbus registers are used to access each parameter in this range. The first register holds the upper 16-bits of the parameter, and the second register holds the lower 16-bits of the parameter. Most Copley parameters are either 16 or 32 bits in size, so this allows most parameters to be easily accessed by reading and writing the corresponding registers in this area.

For 16 bit parameters, the drive will return an error if the value written to the first register is not zero. For parameters larger then 32 bits the two registers corresponding to that parameter will access the first two words of the parameter. Writing to such large parameter using these registers may cause the drive to return an error if the parameter doesn't support partial writes.

Parameters in Copley drives are associated with either RAM (volatile) or flash (non-volatile) memory space. Many parameters have values in both RAM and flash however some parameters are RAM only and some are flash only. This section of registers will access the RAM version of the parameter if the parameter exists in that memory area. If the parameter does not exist in RAM, then the flash version of the parameter will be accessed by these registers.

This memory area consists of 1024 registers. Since there are two registers assigned to every parameter, it allows up to 512 parameters to be accessed (parameters 0 through 511). Not every parameter in this area actually exists in the firmware. Reading the registers associated with a parameter that does not exist in the drive will return zero. Writes to non-existing parameters will cause the drive to return an

error.

To convert a parameter ID into the register address used to access it through this area simply multiply the ID value by 2. For example, to read the motor actual position (parameter 0x17) you would read register 0x2E for the upper 16-bits and 0x2F for the lower 16 bits.

Flash parameter access

Modbus registers in the range 0x0800 through 0x0BFF are mapped to parameters in exactly the same way as registers 0x0000 through 0x03FF. The main difference is that parameters in this range always target the flash version of the parameter. Writing to a parameter which doesn't exist or doesn't exist in flash memory will cause an error.

To convert a parameter ID to the address of the first register in this area, multiply the ID by 2 and add 0x0800.

Long parameter access

Modbus registers in the range 0x1000 through 0x107F can be used to read/write any drive parameter, including those parameters which are longer then 32-bits.

To write a parameter using this memory area, use the 'Write Multiple Registers' Modbus command to write a group of registers starting at address 0x1000. The values written to these registers should start with the parameter ID, followed by the values to be written to the parameter. These registers must all be written as a single Modbus command.

For example, drive parameter 0x70 is used to configure the first general purpose output of the drive. General purpose output parameters consist of a 16-bit value followed by up to two 32-bit values depending on the function being programmed to the output. For example, output configuration 4 sets the output when the motor position is between two values which are sent as the two 32-bit parameters. To configure output 0 to be active when the position is between 0x1234567 and 0x23456789 you would write the following 6 values to Modbus registers 0x1000 – 0x1005 using a single write:

0x0070, 0x0004, 0x1234, 0x5678, 0x2345, 0x6789

To read a large parameter using this register bank, first write the parameter ID to 0x1000. If only register 0x1000 is written then this will not cause the parameter to be set, so for a read only register 0x1000 should be written. Then read the value of the parameter from this bank of registers.

The parameter ID will be returned in register location 0x1000. The number of 16-bit words of valid data will be returned in 0x1001, and the parameter value will be returned starting with 0x1002. For example, after the write to parameter 0x70 described above reading from registers starting with 0x1000 would return the following values:

0x0070, 0x0005, 0x0004, 0x1234, 0x5678, 0x2345, 0x6789

The first value is the parameter ID, the second is the number of words of data and the remainder is contents of the parameter. If more then 7 registers had been read then the remaining register values would be filled with zeros.

Any read within this register area will trigger a read of the parameter who's ID was last written to 0x1000. For example, if only the parameter value were of interest then one could read start at 0x1002 and just receive the parameter value.

When reading or writing values using this register space the drive will only access the parameter in the page specified by the parameter ID. Bit 12 of the parameter ID is used to identify which memory space to use. RAM memory will be accessed if bit 12 is clear such as in the example above. Flash will be accessed if bit 12 is set (i.e. 0x1070 would be output 0 configuration from flash). An error will be returned on reads or writes to variables that don't exist.

Command registers:

Modbus registers in the 0x2000 through 0x200F area are used to perform special drive functions. The following registers are defined in this area:

Register	Function
0x2000	This register is used to perform a trajectory function. The value written to the register determines which function is performed. The values written are the same as the values written using the trajectory op-code over the serial port. 0 Abort any trajectory currently in progress 1 Start a new move or update the move parameters if a move is currently in progress. 2 Start a homing operation 3 Save the move parameters (destination, max velocity, etc) into a queue used for multi-segment moves. 4 Start a multi-segment move based on the segments saved using value 3. The value read from this register is reserved for future use and should be ignored.
0x2001	Start/stop a CVM program running. To start a CVM program, write the program number to this register. To stop a CVM program write -1 (0xFFFF). Reading this register returns the program number of the currently running CVM program.
0x2002	Configure a CVM program to run at power-up by writing the program number to this register. Write -1 to prevent any CVM program from running at startup. Reading this register returns the program number configured to run at startup.
0x2003	Reading this register returns the current CVM status value. Writes are not supported.
0x2004 – 0x200F	These registers are reserved for future use.

Indexer register access

Modbus registers in the range 0x4000 through 0x403F are used to access the 32 32-bit Indexer registers. To access indexer register N read/write the upper 16-bits from address 2*N + 0x4000, and the lower 16-bits from address 2*N + 0x4001.

Revision History

Date	Version	Revision
7/11/2019	Rev 00	Initial release