

Title: Data format and Endianness in MPiec Controllers

Product(s): MP3200iec, MP2300iec, MP2600iec,
MP3300iec

Doc. No. TN.MPIEC.02

Overview

When using any network protocol such a Modbus to transmit data between two devices, inconsistencies in methods of organizing the data can lead to incorrect values in the data registers.

Bits, Bytes, and Words

Internally, all processors “speak” binary, a series of 1’s and 0’s called **bits** that when taken together represent a piece of data.

Instead of representing bits as only ones or zeros, groups of four bits are represented by a single character (0-9 with values of 10-15 being represented by the letters A through F). This base-16 numbering system is called hexadecimal (or hex). Groups of eight bits form a **byte**. These are represented by two hexadecimal characters. A **word** is made up of two bytes (16 bits).

Representations of the integer 12,345

Bits (binary):	0011 0000 0011 1001
Hexadecimal:	3 0 3 9
Bytes (hex):	30 39

Different data types take up a different amount of data. The most common in Modbus are:

Datatype	Description	Bits	
INT	Integer	16	Also UINT – Unsigned Integer
DINT	Double Integer	32	Also UDINT – Unsigned Double Integer
REAL	Floating point	32	
LREAL	Long Floating point	64	Not supported by all devices.
BOOL	Boolean (bit)	1	
WORD	Word	16	
DWORD	Double Word	32	

Understanding byte order and endianness

When sending data from one device to another, different processors use different formats of sending the bytes. In the example above, some processors will first send the least significant byte (39) and then send the most significant byte (30). This is called “Little Endian” because it’s sending the little end of the data first. Other processors will send the

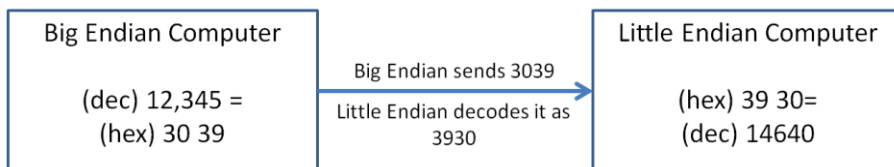
Title: Data format and Endianness in MPiec Controllers

Product(s): MP3200iec, MP2300iec, MP2600iec, MP3300iec

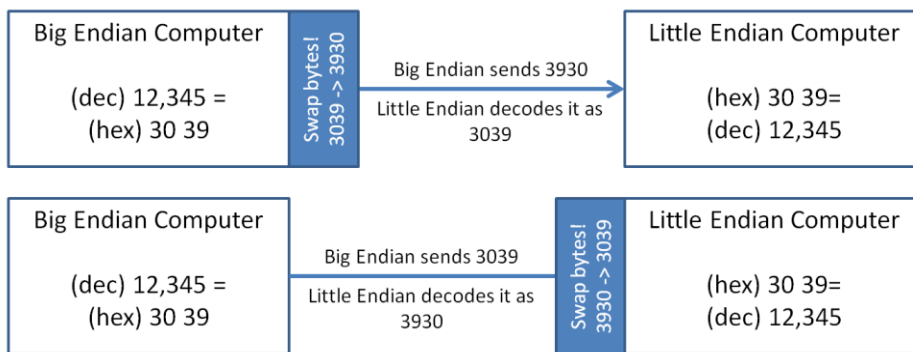
Doc. No. TN.MPIEC.02

data in the opposite order. First 30 and then 39. This is called “Big Endian” because it’s starting at the big end.

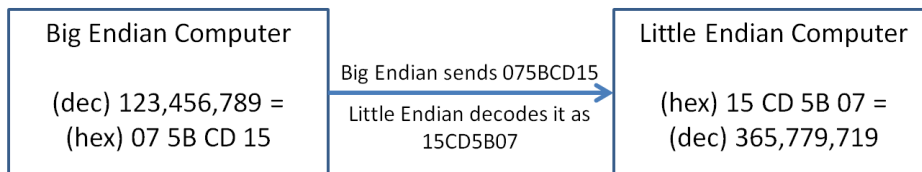
If one device formats its data with big endian and another with little endian, then the data being transmitted will not be decoded correctly.



To alleviate this discrepancy, the order that the bytes are read must be swapped so that the two devices are interpreting the data the same way. This can be done on either device.



The above example is for a 16-bit data type, but the same idea holds true for 32-bit data: Big endian sends the big end first, and little endian reads the little end first. The words are swapped and the bytes that make up the word are also swapped.



Title: Data format and Endianness in MPiec Controllers

Product(s): MP3200iec, MP2300iec, MP2600iec, MP3300iec

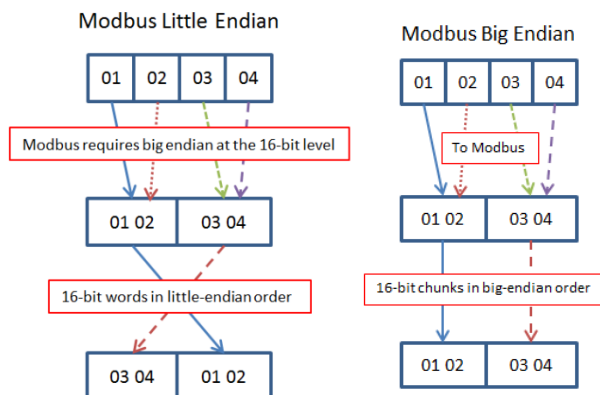
Doc. No. TN.MPIEC.02

Endianness when using Modbus

Modbus defines how data is to be formatted for 16-bit data; it uses the big endian format. It does not, however, define how data is formatted for 32-bit data. Different manufacturers have different methods of sending the data.

A big endian device would send the data the same way. If the number (hex) 01 02 03 04 is sent over Modbus, the byte order would be the same as before: First 01, then 02, then 03, then 04.

A little endian device would normally send 04 03 02 01, but this is not how it works over Modbus. Modbus specifications dictate that the individual 16-bit segments (words) are in big endian order, but then those big endian words are sent in little endian order. This creates a hybrid format.



How endianness is handled by MPiec Controllers

Different models of MPiec controllers have different endianness:

Controller	PLC Runtime	Endianness
MP2300Siec / MP2310iec	ProConOS	Little endian
MP2600iec	eCLR	Little endian
MP3200iec / MP3300iec	eCLR	Big endian

MPiec controllers allow you to select different types of registers in which to store data. When using %I and %Q registers, endian translations were handled automatically by the controller based on data type. Byte swapping was sometimes still necessary but it was generally an easy fix. The disadvantage of these types of registers is that they can only be written to by one of the devices. If two or more devices needed to make changes to the same register, the user had to manually configure multiple variables and keep track of which were the valid values.

Title: Data format and Endianness in MPiec Controllers

Product(s): MP3200iec, MP2300iec, MP2600iec, MP3300iec

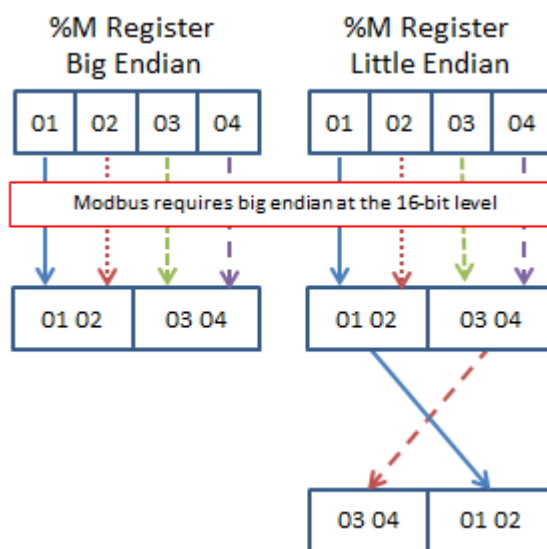
Doc. No. TN.MPIEC.02

MotionWorks IEC version 3.x introduced %M registers for Modbus. A %M register allows for reading or writing to the same register by multiple devices. Because both big and little endian devices can write to these registers, it is not possible to automatically handle endian conversions.

Since Modbus specifies the format of a 16-bit word, either big or little endian controllers will swap bytes within a word. How they read or write words to Modbus will differ, however, based on endianness.

- Big endian controllers will write the high word first.
- Little endian controllers will write the low word first.

Often, other devices such as HMIs are also writing to and reading from the %M registers. They could be using big endian, little endian, or a hybrid format. As a result, it can be a challenge to get data transmitting correctly.



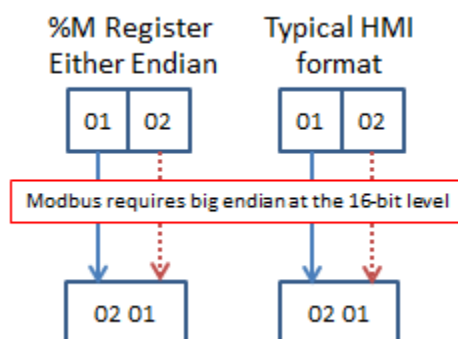
Title: Data format and Endianness in MPiec Controllers

Product(s): MP3200iec, MP2300iec, MP2600iec, MP3300iec

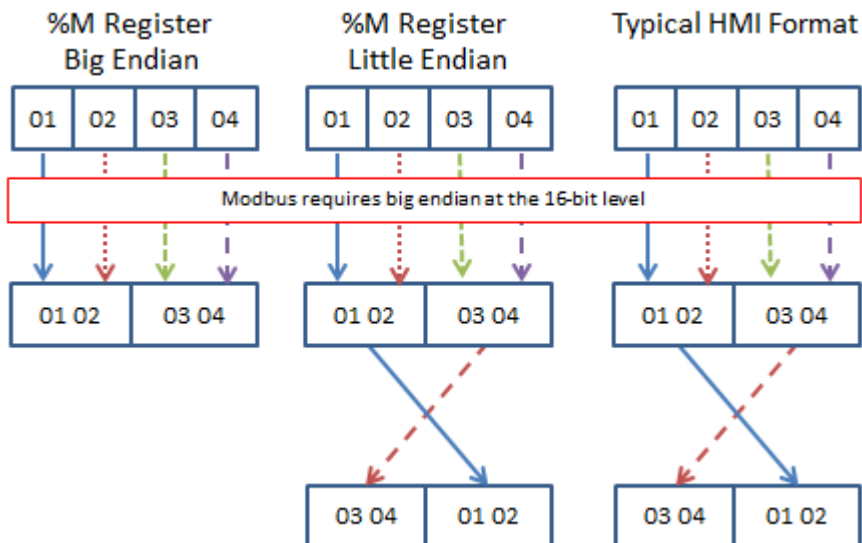
Doc. No. TN.MPIEC.02

Word Swapping between devices

16-Bit Data Types: Since 16-bit data format is specified by Modbus, there is no need to swap bytes regardless of endianness of the controllers.



32-bit Data Types: The Modbus specification requires 16-bit words to be big endian, but the word order is not specified by the specification. Typically, big endian controllers require a word swap in the HMI, but little endian controllers do not.



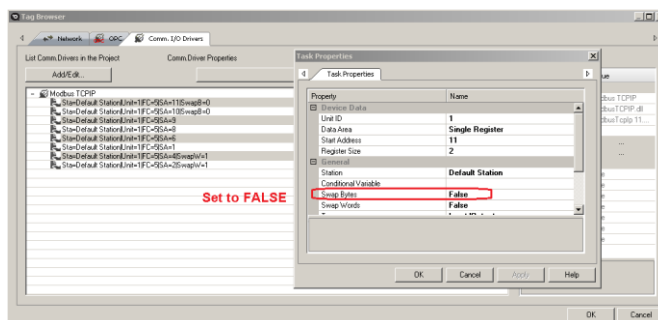
Title: Data format and Endianness in MPiec Controllers

Product(s): MP3200iec, MP2300iec, MP2600iec, MP3300iec

Doc. No. TN.MPIEC.02

Appendix 1: Communicating with %M registers with VIPA Movicon (v11.4.1151)

VIPA's Movicon software gives users the ability to swap bytes and swap words when communicating via Modbus. This is done in the *Task Properties* settings. The *Task Properties* can be opened via the "Comm I/O Drivers" in the *Tag Browser*, and right-clicking a tag to edit its properties. The properties that do this are called *Swap Bytes* and *Swap Words*.



Below are the appropriate settings for these selections based on controller and data-type:

Controller Type	16-bit tag	32-bit tag	64-bit tag
Big Endian (MP3200iec/MP3300iec)	Swap Bytes: True Swap Words: N/A	Swap Bytes: True Swap Words: True	N/A ⁴
Little Endian (MP3200iec/MP3300iec)	Swap Bytes: True Swap Words: N/A	Swap Bytes: True Swap Words: False	Swap Bytes: True Swap Words: False

¹Modbus determines correct byte order, so Swap Bytes is not changeable for 8-bit data types.

²Writes to lower byte of the Modbus word

³Writes to upper byte of the Modbus word

⁴This cannot be accomplished via the HMI software. Use IEC byte swapping function blocks.

Title: Data format and Endianness in MPiec Controllers

Product(s): MP3200iec, MP2300iec, MP2600iec, MP3300iec

Doc. No. TN.MPIEC.02

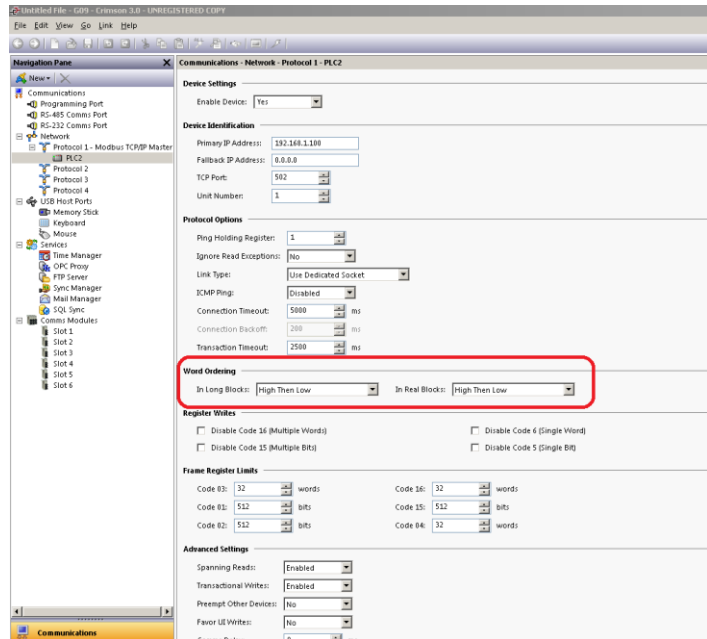
Appendix 2: Communicating with %M registers with Red Lion Crimson (v3.0)

Red Lion's HMI programming software, Crimson, has two different places where byte-swapping can be accomplished.

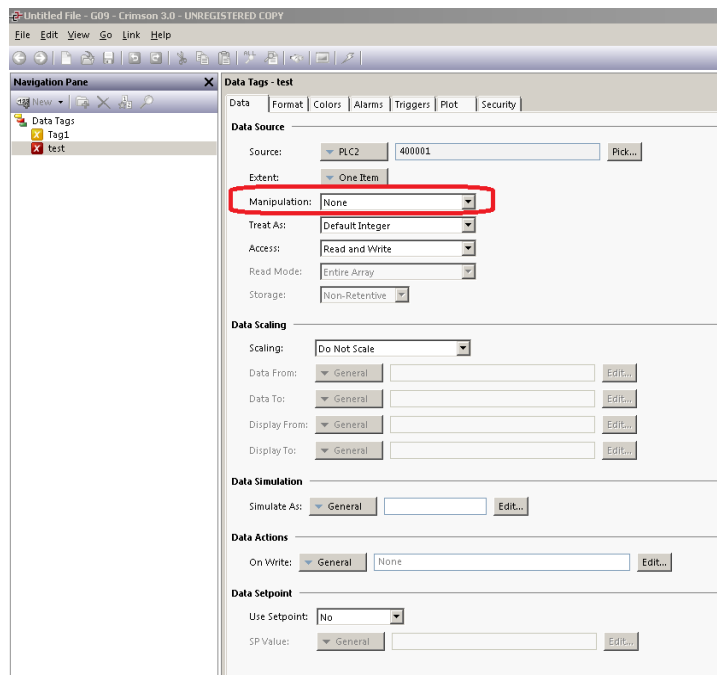
The first is in the ModBus communication configuration settings. This sets the default word order for all variable types.

The two relevant fields are in the *Word Ordering* section:

1. *In Long Blocks* is used for 32-bit non-floating-point variables
2. *In Real Blocks* is used for 32-bit floating-point variables



The second place where word order can be set is in the individual tag settings. The *Manipulation* field indicates whether the data is sent in the default byte order (as set above) for a particular tag or whether it is to be reversed.



Title: Data format and Endianness in MPiec Controllers

Product(s): MP3200iec, MP2300iec, MP2600iec,
MP3300iec

Doc. No. TN.MPIEC.02

Appendix 2 (cont.): Communicating with %M registers with Red Lion Crimson

Below are the appropriate settings for these selections based on controller and data-type:

Controller Type	16-bit tag ¹	32-bit tag	64-bit tag
Big Endian (MP3200iec/MP3300iec)	<i>Word Ordering:</i> N/A <i>Manipulation:</i> None	<i>Word Ordering</i> ² : High Then Low (both) <i>Manipulation:</i> None	Not supported by Red Lion
Little Endian (MP3200iec/MP3300iec)	<i>Word Ordering:</i> N/A <i>Manipulation:</i> None	<i>Word Ordering</i> ³ : Low then High (both) <i>Manipulation:</i> None	Not supported by Red Lion

¹ *Word Ordering* settings only apply to 32-bit tags.

² *Word Ordering* of “Low then High” with *Manipulation* of “Reverse Words” will also work.

³ *Word Ordering* of “High then Low” with *Manipulation* of “Reverse Words” will also work.

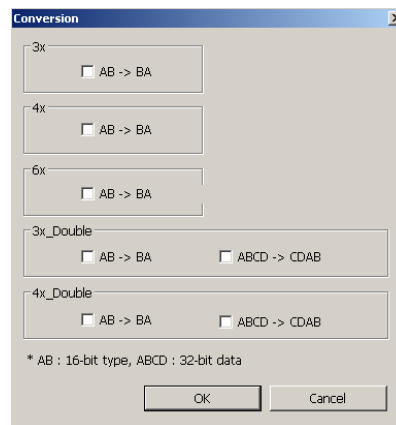
Title: Data format and Endianness in MPiec Controllers

Product(s): MP3200iec, MP2300iec, MP2600iec, MP3300iec

Doc. No. TN.MPIEC.02

Appendix 3: Communicating with %M registers with Maple Systems EZWarePlus (v4.10.07)

Maple Systems' EZWarePlus HMI programming software manipulates byte order in the *Conversion* properties in the PLC Device Properties dialog.



[Edit→System Parameters→Device→(Select PLC) →Settings →(In Device Properties, choose Conversion)]

Below are the appropriate settings for these selections based on controller and data-type:

Controller Type	16-bit tag	32-bit tag	64-bit tag
Big Endian (MP3200iec/MP3300iec)	No byte swapping needed	Set ABCD->CDBA under 4x_Double to TRUE	Not supported by Maple
Little Endian (MP3200iec/MP3300iec)	No byte swapping needed	Set all swapping to FALSE	Not supported by Maple