WM30-WM40

ETHERNET/IP PROTOCOL

Internal version
rev. 1.1

May 20th, 2015

Index

1. COMMUNICATION PROTOCOL 3

1.1 Introduction 3
1.2 System Overview 3
1.3 Data Types 4
1.4 Endian Issues 4
1.5 Geometric representation 4
1.6 Maximum and minimum electrical values 5

2. ETHERNET/IP REQUIRED OBJECTS 6

2.1 Identity Object 6
2.1.1 Class Attributes 6
2.1.2 Instance Attributes 6
2.1.3 Semantics 6
2.1.4 Common Services 7
2.1.5 Get Attribute All Service 7
2.1.6 Reset Service 7
2.1.7 Get Attribute Single Service 8
2.1.8 Request Parameters 8
2.1.9 Identity Object Instances 8
2.1.10 Behavior 9
2.2 Message Router Object 9
2.2.1 Service Request 9
2.2.2 Service Response 9
2.3 Connection Manager Object 9
2.4 Unconnected Message Manager (UCMM) 10
2.5 TCP/IP Interface Object 11
2.5.1 Class Attributes 11
2.5.2 Instance Attributes 11
2.5.3 Common Services 11
2.6 Ethernet Link Object 12
2.6.1 Class Attributes 12
2.6.2 Instance Attributes 12
2.6.3 Common Services 12

3. ETHERNET/IP APPLICATION OBJECTS 13

3.1 Assembly Object 13
3.1.1 Connections 13
3.1.2 Class Attributes 13
3.1.3 Instance Attributes 13
3.1.4 Common Services 13
3.1.5 Assembly Object Instances 14
3.1.6 Behavior 18

4. TECHNICAL NOTES 19

4.1 Parameters 19

4.2 Modbus TCP 19
4.3 ACD 19
4.4 TCP/IP port 19
5. DOCUMENT REVISIONS 20

1. COMMUNICATION PROTOCOL

1.1 Introduction

For a complete description of the EtherNet/IP please refer ODVA site (www.odva.org).

1.2 System Overview

A complete EtherNet/IP WM30/WM40 system will consist of a WM30/WM40 host controller and an EtherNet/IP module, each containing their own microprocessor. The WM30/WM40 will execute power monitoring application firmware, and the EtherNet/IP module will implement an EtherNet/IP Adapter class protocol stack. The two devices will communicate with each other using dual port RAM and an appropriate protocol. The following diagram depicts the WM30/WM40's overall system architecture.

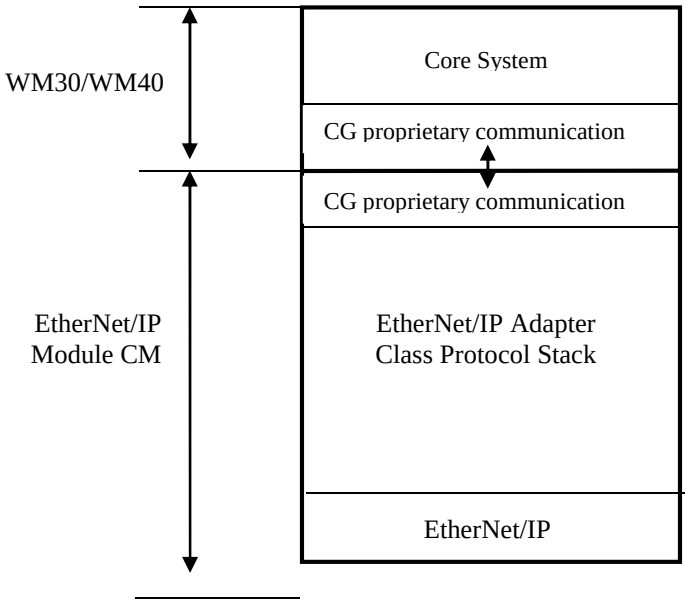


Figure 1-1- System Architecture

1.3 Data Types

The following data types are defined within the code and used throughout this document:

- BYTE 8 bit unsigned value
- USINT 8 bit unsigned value
- WORD 16 bit unsigned value
- UINT 16 bit unsigned value
- INT 16 bit signed value
- DWORD 32 bit unsigned value
- UDINT 32 bit unsigned value
- ULINT 64 bit unsigned value
- DINT 32 bit signed value
- FLOAT 32 bit floating point value (IEEE754)

The IEEE754 representation of a 32-bit floating-point number as an integer is defined as follows:

32-bit floating-point

Bits		
31	30 ... 23	22 ... 0
Sign	Exponent	Mantissa

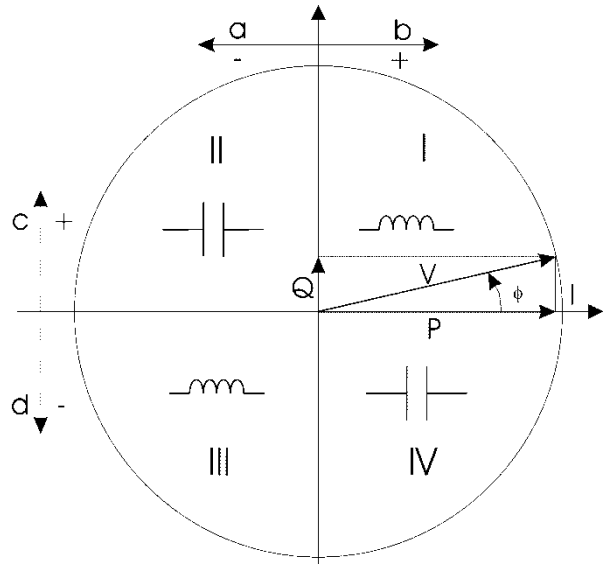
$$(-1)^{sign} * 2^{(Exponent-127)} * 1.Mantissa$$

1.4 Endian Issues

Data format for the EtherNet/IP Encapsulation Protocol in this instrument is *Little-Endian.* Any endian issues arising from differences between the WM30/WM40 and CM hardware will be resolved within the WM30/WM40.

1.5 Geometric representation

According to the signs of the power factor, the active power P and the reactive power Q, it is possible to obtain a geometric representation of the power vector, as indicated in the drawing below, according to EN 62053:



a = Exported active power
b = Imported active power
c = Imported reactive power
d = Exported reactive power

Fig. 1 : Geometric Representation

1.6 Maximum and minimum electrical values

The max and min electric values for each variable are indicated in the following table:

AV4: 400/690VLL AC, 1(2)A
VLN : 160 V to 480VLN
VLL : 277 V to 830VLL

AV5: 400/690VLL AC, 5(6)A
VLN : 160 V to 480VLN
VLL : 277 V to 830VLL

AV6: 100/208VLL AC, 5(6)A
VLN : 40 V to 144VLN
VLL : 70 V to 250VLL

AV7: 100/208VLL AC, 1(2)A
VLN : 40 V to 144VLN
VLL : 70 V to 250VLL

2. EtherNet/IP Required Objects

2.1 Identity Object

Class Code: 01 hex

The Identity Object is used to provide identification information about the device. Each node shall support at least one instance of the identity object. It shall be used by applications wishing to determine which nodes are on the network.

See Section 5-2 of the *CIP Common Specification* for full details of this object.

2.1.1 Class Attributes

Attr ID	Access	Name	Data Type	Default Value
0x01	Get	Revision	UINT	1
0x02	Get	Max Instance	UINT	1
0x03	Get	Number of Instances	UINT	1
0x06	Get	Maximum ID Number Class Attributes	UINT	7
0x07	Get	Maximum ID Number Instance Attributes	UINT	7

2.1.2 Instance Attributes

Attr ID	Access	Name	Data Type
0x01	Get	Vendor ID	UINT
0x02	Get	Device Type	UINT
0x03	Get	Product Code	UINT
0x04	Get	Revision	Struct of:
		Major Revision	USINT
		Minor Revision	USINT
0x05	Get	Status	WORD
0x06	Get	Serial Number	UDINT
0x07	Get	Product Name	SHORT_STRING

2.1.3 Semantics

The Identity Object's attributes are defined as follows:

Vendor ID – Vendor IDs uniquely identify a particular vendor. The Carlo Gavazzi vendor ID is 1303 (decimal).

Product Code – The vendor assigned Product Code shall identify a particular product within a device type. Each vendor shall assign this code to each of its products. The Product Code typically maps to one or more catalogue/model numbers. Products shall have different codes if their configuration and or runtime options *are* different. The value zero is not valid.

Revision – The Revision attribute, which consists of Major and Minor Revisions, shall identify the Revision of the item the Identity Object is representing. The value zero shall not be valid for either the Major or Minor Revision fields. The Major and Minor Revision are typically displayed as major.minor. Minor revisions shall be displayed as three digits with leading zeros as necessary. The Major Revision attribute shall be limited to 7 bits. The eighth bit is reserved and shall be zero.

The major revision shall be incremented by the vendor when there is a significant change to the "fit, form, or function" of the product. Any changes that affect the configuration choices available to the user (and therefore the EDS file) require incrementing the major revision.

The Minor revision may be used to identify changes in a product that do not affect user configuration choices, e.g. bug fixes, hardware component change, labeling change, etc.

Status – This attribute shall represent the current status of the entire device. Its value changes as the state of the device changes. The Status attribute is a WORD. The following table details the status WORD.

Status Values

Bit(s)	Name	Name
0	Owned	A 1 (TRUE) shall indicate that the device has an owner.
1		Reserved, set to 0.
2	Configured	A 1 (TRUE) shall indicate the application of the device has been configured to do something different than the “out-of-box” default. This shall not include configuration of the communications.
3		Reserved, set to 0.
4,5,6,7		Vendor-Specific.
8	Minor Recoverable Fault	A 1 (TRUE) shall indicate the device detected a problem with itself, which is thought to be recoverable. The problem shall not cause the device to go into one of the faulted states.
9	Minor Unrecoverable Fault	A 1 (TRUE) shall indicate the device detected a problem with itself, which is thought to be unrecoverable. The problem shall not cause the device to go into one of the faulted states.
10	Major Recoverable Fault	A 1 (TRUE) shall indicate the device detected a problem with itself, which caused the device to go into the “Major Recoverable Fault” state.
11	Major Unrecoverable Fault	A 1 (TRUE) indicates the device detected a problem with itself, which caused the device to go into the “Major Unrecoverable Fault” state.
12,13		Reserved, set to 0.
14,15		Reserved, set to 0.

Serial Number – This attribute provides a number used in conjunction with the Vendor ID to form a unique identifier for each device. Each vendor shall be responsible for guaranteeing the uniqueness of the serial number across all its devices.

Product Name – This test string may represent a short description of the product/product family represented by the product code in attribute 3. The same product code may have a variety of product name strings. The maximum number of characters in this string shall be 32.

2.1.4 Common Services

Service Code	Class/Instance Usage	Service Name
0x01	Class/Instance	Get_Attributes_All
0x05	Instance	Reset
0x0E	Class/Instance	Get_Attribute_Single

2.1.5 Get Attribute All Service

The Get Attribute All service will return a concatenation of all class or instance attributes.

2.1.6 Reset Service

When the Identity Object receives a Reset request it will determine if the reset type is supported, respond to the request and execute the type of reset requested. The Reset Service 1 will restore the factory default settings for ACD.
The service accepts the following request parameter:

Request Parameters

Parameter	Data Type	Description
Type	USINT	Type of Reset to perform

The type of Reset requested may be any of the following:

Reset Types

Value	Name	Description
0	Power-Cycle	Emulate as closely as possible cycling power on the item the Identity Object represents.
1	Power-Cycle	Emulate as closely as possible cycling power on the item the Identity Object represents.
2 – 255	Reserved	Reserved

2.1.7 Get Attribute Single Service

The Get Attribute Single service will return the single attribute specified by the parameter *Attribute ID*.

2.1.8 Request Parameters

Parameter	Data Type	Description
Attribute ID	UINT	Identifies the attribute to be read / returned

2.1.9 Identity Object Instances

Besides the class instance (instance 0), the CM's Identity Object will support the following instance.

2.1.9.1 Instance 1 (WM30/WM40+CM)

Instance 1 will represent the WM30/WM40 device with its EtherNet/IP module.

Instance 1 of the Identity Object will report the following instance specific attribute values:

Instance 1 Attribute Values

Attr ID	Name	Value
0x01	Vendor ID	1303 (Gavazzi Controls)
0x02	Device Type	0x64 (100d)
0x03	Product Code	1
0x04	Revision	4.2
0x05	Status	*Dynamic*
0x06	Serial Number	*See schema below*
0x07	Product Name	a) "WM30 with MCEI" b) "WM40 with MCEI" c) "WM40 with MCEIM"

Product Name: a), b) or c) depending by base and module connected

Serial number as the base. Char B is omitted. Letter Year is replaced by number.

Gavazzi Serial Number format

Y	year	D	D	D	X	X	X	X	0	0	1	1

Y = production site (B for Belluno)
Year = A (2001), B (2002), C (2003)… L (2012)…
DDD = day of the year
xxxx = number piece of the day

For example: B L 365 1234 0011 becomes 12 365 1234

2.1.10 Behavior

The behavior of the Identity Object with regards to the Get Attributes All service is typical of any other CIP object. Its handling of the reset service however, is considerably more implementation specific. The CM's Identity Object instance will handle reset services in the following manner regardless of the reset type requested:

- **Instance 1 (WM30/WM40+CM) –** Respond successfully to the request, but continue normal operation.

2.2 Message Router Object

Class Code: 02 hex

The Message Router Object shall provide a messaging connection point through which a client may address a service to any object class or instance residing in the physical device.

The Message Router Object performs the following functions:

- Interprets the Class Instance specified in a message
- Routes a service to the specified object
- Interprets services directed to it
- Routes a response to the correct source

2.2.1 Service Request

Interpretation of the Class Instance is performed on every service received by the Message Router.

Any Class Instance that cannot be interpreted by a device's implementation of a Message Router will report the Object_Not_Found error.

The service is then routed to an application object. The information routed to an application object must include the following:

- The service code of the message
- A pointer to the next byte to access in the message
- The number of bytes left to process in the message
- A device specific identification of the message source or Connection ID for the response

If no Class Instance specification is supplied or if the message specifies the Message Router Object itself, the service is processed by the Message Router.

2.2.2 Service Response

All service responses are routed to the source of the service request. The source could be one of the following:

- The Unconnected Message Manager
- An active connection to the Message Router Object

This requires the service request sources to supply a unique identifier. This identifier is device and implementation specific and simplifies implementation of the Message Router in any given application.

2.3 Connection Manager Object

Class Code: 06 hex

The Connection Manager (ConnM) opens and closes connections as well as maintains the network's connection tables. It is responsible for:

- setting up a connection within its node
- processing all connection requests sent on the local service
- forwarding an open connection service via the UCMM to the next node in a connection path

- establishing a connection to a target node
- receives forward open and forward close service requests

Connection requests are originated by:

- other nodes via the UCMM
- an application on a node

The following steps outline how the Connection Manager of a target node functions upon receiving a connection request by an originating node.

1. The originator's UCMM contacts the target's UCMM with a connection request.
2. The request is routed through the target's Message Router to the Connection Manager.
3. The Connection Manager allocates the needed resources.
4. A connection is made to the originator node.

Connection Tables hold information about all connections in which the node participates.

See Section 3-5 of the *CIP Common Specification* for full details of the Connection Manager Object.

Class Specific Services

Service Code	Class/Instance Usage	Service Name
0x54	Instance	Fwd_Open
0x4E	Instance	Fwd_Close

2.4 Unconnected Message Manager (UCMM)

The UCMM provides an unconnected request/response message service, limited to a single link that supports multiple outstanding messages. The UCMM is required for all nodes, and shall support at least one outstanding message. All incoming requests are forwarded to the message router.

The Message Router (MR) allows an application to open connections to objects within the same node and is responsible for:

- interpreting the part of a message that indicates the local destination object
- routing the message to the appropriate object for execution

The Unconnected Message Manager (UCMM) provides the ability to send a message without an established connection. It is responsible for:

- receiving incoming UCMM messages
- sending and receiving unconnected messages to and from UCMM objects on other nodes
- sending and receiving unconnected messages to and from its local Message Router

NOTE: Once a connection is established, objects such as the MR, UCMM and ConnM are no longer required. Data can go directly to the destination object.

A connected message assumes previously negotiated resources and parameters at its source, its destination(s), and any intermediate transit points. These resources are referenced by a unique connection identifier and do not need to be contained in each message, only the connection ID is needed to identify the message and refer to its related parameters.

An unconnected message provides a means to communicate without previously negotiated resources at the destination so it shall carry full destination ID details, internal data descriptors and full source ID details if a reply is requested. Unconnected messages are used to create connections.

2.5 TCP/IP Interface Object

Class Code: F5 hex

The TCP/IP Interface Object provides access to a device's TCP/IP network interface information.

See Volume 2, Section 5-3 of *EtherNet/IP Adaptation of CIP* for full details of this object.

2.5.1 Class Attributes

Attr ID	Access	Name	Data Type	Default Value
0x01	Get	Revision	UINT	2
0x02	Get	Max Instance	UINT	1
0x03	Get	Number of Instances	UINT	1
0x04	Get	Optional Attributes List	STRUCT of UINT ARRAY of UINT	 0x0002, {0x000A, 0x000B}
0x06	Get	Maximum ID Number Class Attributes	UINT	7
0x07	Get	Maximum ID Number Instance Attributes	UINT	11 (0xB)

2.5.2 Instance Attributes

Attr ID	Access	Name	Data Type	Description
0x01	Get	Status	UDINT	See Volume 2, Section 5-3.2.2 of the *EtherNet/IP Adaptation of CIP* document for details.
0x02	Get	Configuration Capability	UDINT	
0x03	Get	Configuration Control	UDINT	
0x04	Get	Physical Link Object	Structure	
0x05	Get	Interface Configuration	Structure	
0x06	Get	Host Name	STRING	
0x0A	Set	SelectAcd	Bool	
0x0B	Set	LastConflictDetected	Structure	

2.5.3 Common Services

Service Code	Class/Instance Usage	Service Name
0x01	Class/Instance	Get_Attributes_All
0x0E	Class/Instance	Get_Attribute_Single
0x10	Instance	Set_Attribute_Single

2.5.3.1 Get Attribute All Service

The Get Attribute All service will return a concatenation of all class or instance attributes.

2.5.3.2 Get Attribute Single Service

The Get Attribute Single service will return the single attribute specified by the parameter *Attribute ID*.

Request Parameters

Parameter	Data Type	Description
Attribute ID	UINT	Identifies the attribute to be read / returned

2.5.3.3 Single Service

The Set Attribute Single service will set the single attribute specified by the parameter *Attribute ID* to the value specified by the parameter *Attribute Data.*

Request Parameters

Parameter	Data Type	Description
Attribute ID	UINT	Identifies the attribute to be read / returned
Attribute Data	Attribute Specific	Contains the value to which the specified attribute is to be set

2.6 Ethernet Link Object

Class Code: F6 hex

The Ethernet Link Object maintains link specific counters and status information for an Ethernet 802.3 communications interface.

See Volume 2, Section 5-4 of *EtherNet/IP Adaptation of CIP* for full details of this object.

2.6.1 Class Attributes

Attr ID	Access	Name	Data Type	Default Value
0x01	Get	Revision	UINT	1
0x02	Get	Max Instance	UINT	1
0x03	Get	Number of Instances	UINT	1
0x06	Get	Maximum ID Number Class Attributes	UINT	7
0x07	Get	Maximum ID Number Instance Attributes	UINT	3

2.6.2 Instance Attributes

Attr ID	Access	Name	Data Type	Description
0x01	Get	Interface Speed	UDINT	Speed in megabits / sec
0x02	Get	Interface Flags	DWORD	Interface Status Flags
0x03	Get	Physical Address	USINT[6]	MAC Address

2.6.3 Common Services

Service Code	Class/Instance Usage	Service Name
0x01	Class/Instance	Get_Attributes_All
0x0E	Class/Instance	Get_Attribute_Single

2.6.3.1 Get Attributes All Service

The Get Attribute All service will return a concatenation of all class or instance attributes.

2.6.3.2 Get Attribute Single Service

The Get Attribute Single service will return the single attribute specified by the parameter *Attribute ID*.

Request Parameters

Parameter	Data Type	Description
Attribute ID	UINT	Identifies the attribute to be read / returned

3. EtherNet/IP Application Objects

3.1 Assembly Object

Class Code: 04 hex

The assembly object collects attributes from multiple objects, allowing data to or from each object to be sent or received over a single connection. Assembly objects are used to produce and/or consume data to/from the network. An instance of the assembly object can both produce and consume data from the network.
Static assemblies are factory determined and cannot be modified by the user of the product. Members cannot be added or deleted. The implementation of the Assembly object shall be static.

See Section 5-5 of the *CIP Common Specification* for full details of this object.

3.1.1 Connections

The Assembly Object will support both scheduled (Class 1) and unscheduled connections (Class 3 & UCMM.
Both connections are used to access instances 100d to 108d of the Assembly Object.

a.) 9 Class 1 connections (UDP) each to the 9 assemblies 100 to 108 at 100 ms RPI ran stable
b.) 9 Class 3 connections (TCP) each to the 9 assemblies 100 to 108 at 200 ms RPI ran stable

both a.) and b.) can run at the same time

3.1.2 Class Attributes

Attr ID	Access	Name	Data Type	Default Value
0x01	Get	Revision	UINT	2
0x02	Get	Max Instance	UINT	108d
0x03	Get	Number of Instances	UINT	9
0x04	Get	Optional Attributes List	STRUCT of UINT ARRAY of UINT	 0x0001, {0x0004}
0x06	Get	Maximum ID Number Class Attributes	UINT	7
0x07	Get	Maximum ID Number Instance Attributes	UINT	4

3.1.3 Instance Attributes

Attr ID	Access	Name	Data Type	Default Value
0x03	Get	Data	Instance Dependant (See section 4.1.6.1)	All of the member data packed into one array.
0x04	Get	Size	Size (in bytes) of the Data attribute	Instance Dependant (See section 4.1.6.1)

3.1.4 Common Services

Service Code	Class/Instance Usage	Service Name
0x0E	Class/Instance	Get Attribute Single

3.1.4.1 Get Attribute Single Service

The Get Attribute Single service will return the single attribute specified by the parameter *Attribute ID*.

Request Parameters

Parameter	Data Type	Description
Attribute ID	UINT	Identifies the attribute to be read / returned

3.1.5 Assembly Object Instances

The CM's Assembly Object will support 9 instances (instances 100d to 108d).

The table below defines the size and content of the Data attribute for each Assembly object instances.

3.1.5.1 Versions and serial number

CIP Assy. Inst. 0x064 (100d)
No. of Elements 10
Data Type Custom
Data Access Read

Modicom address	Physical address	Length (words)	VARIABLE ENG. UNIT	Data Format	Notes	Firmware compatibility
300001	0000h	1	Base firmware version	Custom	MSB: ASCII code for model LSB: numeric number for revision	X0
300002	0001h	1	EtherNet/IP module firmware version	Custom	MSB: ASCII code for model LSB: numeric number for revision	X8
300003	0002h	1	Analogue output module firmware version (if present)	Custom	MSB: ASCII code for model LSB: numeric number for revision	X0
300033	0020h	1	Letter 1 (from SX) Letter 2 (from SX)	Custom	MSB: ASCII code LSB: ASCII code	X0
300034	0021h	1	Letter 3 (from SX) Letter 4 (from SX)	Custom	MSB: ASCII code LSB: ASCII code	X0
300035	0022h	1	Letter 5 (from SX) Letter 6 (from SX)	Custom	MSB: ASCII code LSB: ASCII code	X0
300036	0023h	1	Letter 7 (from SX) Letter 8 (from SX)	Custom	MSB: ASCII code LSB: ASCII code	X0
300037	0024h	1	Letter 9 (from SX) Letter 10 (from SX)	Custom	MSB: ASCII code LSB: ASCII code	X0
300038	0025h	1	Letter 11 (from SX) Letter 12 (from SX)	Custom	MSB: ASCII code LSB: ASCII code	X0
300039	0026h	1	Letter 13 (from SX)	Custom	MSB: ASCII code LSB: empty	X0

NOTE 1. In the following document the firmware letter "X" indicates all versions. The number indicates the firmware revision.

3.1.5.2 Instantaneous variables (voltage and current)

CIP Assy. Inst. 0x065 (101d)
No. of Elements 12
Data Type: Floating point - 32 bit IEEE 754
Data Access Read

Modicom address	Physical address	Length (words)	VARIABLE ENG. UNIT	Data Format	Notes	Firmware compatibility
300081	0050h	2	V L1-N	FLOAT		X0
300083	0052h	2	V L2-N	FLOAT		X0
300085	0054h	2	V L3-N	FLOAT		X0
300087	0056h	2	V L-N $\sum$	FLOAT		X0
300089	0058h	2	V L1-L2	FLOAT		X0
300091	005Ah	2	V L2-L3	FLOAT		X0
300093	005Ch	2	V L3-L1	FLOAT		X0
300095	005Eh	2	V L-L $\sum$	FLOAT		X0
300097	0060h	2	A L1	FLOAT		X0
300099	0062h	2	A L2	FLOAT		X0
300101	0064h	2	A L3	FLOAT		X0
300103	0066h	2	A N	FLOAT		X0

3.1.5.3 Instantaneous variables (power, power factor and frequency)

CIP Assy. Inst. 0x066 (102d)
No. of Elements 18
Data Type Floating point - 32 bit IEEE 754
Data Access Read

Modicom address	Physical address	Length (words)	VARIABLE ENG. UNIT	Data Format	Notes	Firmware compatibility
300105	0068h	2	W L1	FLOAT		X0
300107	006Ah	2	W L2	FLOAT		X0
300109	006Ch	2	W L3	FLOAT		X0
300111	006Eh	2	W ∑	FLOAT		X0
300113	0070h	2	VA L1	FLOAT		X0
300115	0072h	2	VA L2	FLOAT		X0
300117	0074h	2	VA L3	FLOAT		X0
300119	0076h	2	VA ∑	FLOAT		X0
300121	0078h	2	VAR L1	FLOAT		X0
300123	007Ah	2	VAR L2	FLOAT		X0
300125	007Ch	2	VAR L3	FLOAT		X0
300127	007Eh	2	VAR ∑	FLOAT		X0
300129	0080h	2	PF L1	FLOAT	Negative values correspond to lead(C), positive values correspond to lag(L)	X0
300131	0082h	2	PF L2	FLOAT		
300133	0084h	2	PF L3	FLOAT		
300135	0086h	2	PF ∑	FLOAT		
300137	0088h	2	Hz	FLOAT		X0
300139	008Ah	2	Phase sequence	FLOAT	Value +1 corresponds to the L1-L2-L3 sequence, value -1 corresponds to wrong sequence	X0

3.1.5.4 Maximum variables (voltage and current)

CIP Assy. Inst. 0x067 (103d)
No. of Elements 12
Data Type Floating point - 32 bit IEEE 754
Data Access Read

Modicom address	Physical address	Length (words)	VARIABLE ENG. UNIT	Data Format	Notes	Firmware compatibility
300337	0150h	2	Max V L1-N	FLOAT		X0
300339	0152h	2	Max V L2-N	FLOAT		X0
300341	0154h	2	Max V L3-N	FLOAT		X0
300343	0156h	2	Max V L-N ∑	FLOAT		X0
300345	0158h	2	Max V L1-L2	FLOAT		X0
300347	015Ah	2	Max V L2-L3	FLOAT		X0
300349	015Ch	2	Max V L3-L1	FLOAT		X0
300351	015Eh	2	Max V L-L ∑	FLOAT		X0
300353	0160h	2	Max A L1	FLOAT		X0
300355	0162h	2	Max A L2	FLOAT		X0
300357	0164h	2	Max A L3	FLOAT		X0
300359	0166h	2	Max A N	FLOAT		X0

3.1.5.5 Maximum variables (power, power factor and frequency)

CIP Assy. Inst. 0x068 (104d)
No. of Elements 17
Data Type Floating point - 32 bit IEEE 754
Data Access Read

Modicom address	Physical address	Length (words)	VARIABLE ENG. UNIT	Data Format	Notes	Firmware compatibility
300361	0168h	2	Max W L1	FLOAT		X0
300363	016Ah	2	Max W L2	FLOAT		X0
300365	016Ch	2	Max W L3	FLOAT		X0
300367	016Eh	2	Max W ∑	FLOAT		X0
300369	0170h	2	Max VA L1	FLOAT		X0
300371	0172h	2	Max VA L2	FLOAT		X0
300373	0174h	2	Max VA L3	FLOAT		X0
300375	0176h	2	Max VA ∑	FLOAT		X0
300377	0178h	2	Max VAR L1	FLOAT	Negative values correspond to lead(C), positive values correspond to lag(L)	X0
300379	017Ah	2	Max VAR L2	FLOAT		
300381	017Ch	2	Max VAR L3	FLOAT		
300383	017Eh	2	Max VAR ∑	FLOAT		
300385	0180h	2	Max PF L1	FLOAT		X0
300387	0182h	2	Max PF L2	FLOAT		X0
300389	0184h	2	Max PF L3	FLOAT		X0
300391	0186h	2	Max PF ∑	FLOAT		X0
300393	0188h	2	Max Hz	FLOAT		X0

3.1.5.6 DMD variables (voltage and current)

CIP Assy. Inst. 0x069 (105d)
No. of Elements 12
Data Type Floating point - 32 bit IEEE 754
Data Access Read

Modicom address	Physical address	Length (words)	VARIABLE ENG. UNIT	Data Format	Notes	Firmware compatibility
300849	0350h	2	DMD V L1-N	FLOAT		X0
300851	0352h	2	DMD V L2-N	FLOAT		X0
300853	0354h	2	DMD V L3-N	FLOAT		X0
300855	0356h	2	DMD V L-N ∑	FLOAT		X0
300857	0358h	2	DMD V L1-L2	FLOAT		X0
300859	035Ah	2	DMD V L2-L3	FLOAT		X0
300861	035Ch	2	DMD V L3-L1	FLOAT		X0
300863	035Eh	2	DMD V L-L ∑	FLOAT		X0
300865	0360h	2	DMD A L1	FLOAT		X0
300867	0362h	2	DMD A L2	FLOAT		X0
300869	0364h	2	DMD A L3	FLOAT		X0
300871	0366h	2	DMD A N	FLOAT		X0

3.1.5.7 DMD variables (power, power factor and frequency)

CIP Assy. Inst. 0x06A (106d)
No. of Elements 17
Data Type Floating point - 32 bit IEEE 754
Data Access Read

Modicom address	Physical address	Length (words)	VARIABLE ENG. UNIT	Data Format	Notes	Firmware compatibility
300873	0368h	2	DMD W L1	FLOAT		X0
300875	036Ah	2	DMD W L2	FLOAT		X0
300877	036Ch	2	DMD W L3	FLOAT		X0
300879	036Eh	2	DMD W ∑	FLOAT		X0
300881	0370h	2	DMD VA L1	FLOAT		X0
300883	0372h	2	DMD VA L2	FLOAT		X0
300885	0374h	2	DMD VA L3	FLOAT		X0
300887	0376h	2	DMD VA ∑	FLOAT		X0
300889	0378h	2	DMD VAR L1	FLOAT	Negative values correspond to lead(C), positive values correspond to lag(L)	X0
300891	037Ah	2	DMD VAR L2	FLOAT		
300893	037Ch	2	DMD VAR L3	FLOAT		
300895	037Eh	2	DMD VAR ∑	FLOAT		
300897	0380h	2	DMD PF L1	FLOAT		X0
300899	0382h	2	DMD PF L2	FLOAT		X0
300901	0384h	2	DMD PF L3	FLOAT		X0
300903	0386h	2	DMD PF ∑	FLOAT		X0
300905	0388h	2	DMD Hz	FLOAT		X0

3.1.5.8 Total and partial (tariff) energy meters

CIP Assy. Inst. 0x06B (107d)
No. of Elements 18
Data Type Floating point - 32 bit IEEE 754
Data Access Read

EtherNet/IP

Length (words)	VARIABLE ENG. UNIT	Data Format	Notes
2	Total KWh+	FLOAT	**EXAMPLE** A large energy value could be displayed as 123,456,789,234.567 kWh where 123,456 is the GWh metering result and 789,234.567 is the kWh metering result.
2	Total GWh+	FLOAT	
2	Total Kvarh+	FLOAT	
2	Total Gvarh+	FLOAT	
2	Total KWh-	FLOAT	
2	Total GWh-	FLOAT	
2	Total Kvarh-	FLOAT	
2	Total Gvarh-	FLOAT	
2	Partial KWh+	FLOAT	
2	Partial GWh+	FLOAT	
2	Partial Kvarh+	FLOAT	
2	Partial Gvarh+	FLOAT	
2	Partial KWh-	FLOAT	
2	Partial GWh-	FLOAT	
2	Partial Kvarh-	FLOAT	
2	Partial Kvarh-	FLOAT	
2	Hours counter	FLOAT	
2	Minutes counters	FLOAT	**Values are limited from 0 to 59**

Modbus TCP

Modicom address	Physical address	Length (words)	VARIABLE ENG. UNIT	Data Format	Notes	Firmware compatibility
301281	0500h	4	Total KWh+	ULINT	Values in Wh or varh	X0
301285	0504h	4	Total Kvarh+	ULINT		X0
301289	0508h	4	Total KWh-	ULINT		X2
301293	050Ch	4	Total Kvarh-	ULINT		X2
301297	0510h	4	Partial KWh+	ULINT		X0
301301	0514h	4	Partial Kvarh+	ULINT		X0
301305	0518h	4	Partial KWh-	ULINT		X2
301309	051Ch	4	Partial Kvarh-	ULINT		X2
301313	0520h	4	Hours counter	ULINT	Hours value: integer part got from the division of the counter by 100 Minutes value: rest of the previous computation (decimal part)	X0

3.1.5.9 Alarm and output status

CIP Assy. Inst. 0x06C (108d)
No. of Elements 2
Data Type Bitwise
Data Access Read

Modicom address	Physical address	Length (words)	VARIABLE ENG. UNIT	Data Format	Notes	Firmware compatibility
316385	4000h	1	Virtual alarm	Bitwise	Bit value: 0 = OFF Bit value: 1 = ON Bit position (LSB concept): 0: Virtual alarm 1 1: Virtual alarm 2 2: Virtual alarm 3 3: Virtual alarm 4	X0
316386	4001h	1	Output (port)	Bitwise	Bit value: 0 = OFF Bit value: 1 = ON (Note: only if the port is not linked to the counter) Bit position (LSB concept): 0: Port1 1: Port2	X0

3.1.5.10 Heartbeat Instances

In addition to the Assembly instances shown above, the CM will also recognize two heartbeat instances. A heartbeat instance is a virtual output instance which can be specified by devices wishing to establish **Input Only** and **Listen Only** Class 1 I/O connections to the CM. Data can neither be read from, nor written to, the heartbeat instance. It is merely a programming construct which serves to keep the connection alive.

- The CM's heartbeat instances will be **98** for the **Input Only** connection and **99** for the **Listen Only** connection

3.1.6 Behavior

The purpose of the Assembly Object is to act as a network interface to the WM30/WM40's data. That data can be accessed by a variety of means: Class 1 or Class 3 connections and also with UCMM messages.

4. Technical notes

4.1 Parameters

For EtherNet/IP Communication Module, the user could be set the following parameters from Base menu:

- IP addressed
- Subnet
- Gateway
- Modbus TCP port
- Address Conflict Detect (ACD)

4.2 Modbus TCP

The communication module supports Modbus TCP. The protocol is the same of the WM30/WM40 with RS485 (Modbus RTU). Please see the document *"WM30-WM40 CP V2.8 November 2013"* for protocol details. In this product, only 1 Modbus TCP connection (1 socket) is allowed.
ETH/IP and MODBUS TCP are always enabled and they could be run at the same time, however it's recommended to use ETH/IP without Modbus because the full conformance tests have been carried out without active Modbus communication.

4.3 ACD

If ACD is detected from communication module, then the base module will show "ACD Found" on display. It is necessary to check the network configuration and after the problem is resolved, switch off/switch on the instrument to re-establish the communication.

4.4 TCP/IP port

Default EtherNet/IP Port:

- **UDP (implicit message): 2222 (0x08AE)**
- **TCP (explicit message): 44818 (0xAF12)**

Default Modbus TCP Port: 502 (0x01F6)

Only Modbus Port could be set by the user.

5. Document Revisions

Rev 1.1	Table “Instance 1 Attribute Values”: Changed Revision from 4.1 to 4.2. 3.1.5.1 Versions and serial number: Removed all references to special models.