


# MODBUS TCP FOR H0/H2/ H4-ECOM100

---

# CHAPTER 5



## In This Chapter...

Modbus TCP .....	5-2
Supported Modbu Function Codes .....	5-4
Network Server (Slave) Operation .....	5-5
Network Client (Master) Operation.....	5-15
H0/H2/H4 -ECOM100 System Memory .....	5-30

# Modbus TCP

Modbus TCP is essentially the serial Modbus RTU protocol encapsulated in a TCP/IP wrapper. Modbus RTU is used for serial communications between a master and slave(s) devices. Modbus TCP is used for TCP/IP communications between client and server devices on an Ethernet network. The TCP version of Modbus follows the OSI Network Reference Model.



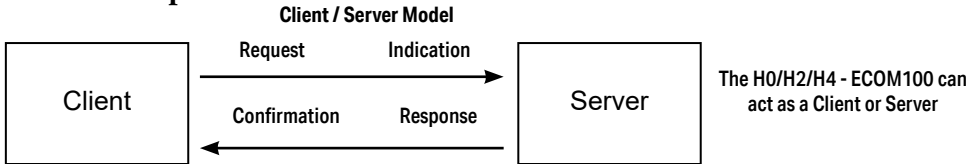
**NOTE:** You must configure the peer to peer configuration as shown in Chapter 3 (page 3-10) to use Modbus TC/P. If the peer to peer configuration is not configured the ECOM100 will use Host Protocol instead of Modbus TC/P.

## Client / Server Model

The Modbus messaging service provides a Client/Server communication between devices connected on an Ethernet TCP/IP network. This client / server model is based on four types of messages:

- Modbus Request - the message sent on the network by the Client to initiate a transaction
- Modbus Confirmation - the Response Message received on the Client side
- Modbus Indication - the Request message received on the Server side
- Modbus Response - the Response message sent by the Server

## Protocol Description



A typical Modbus TCP frame consists of the following fields:

The **MBAP header** (Modbus Application Protocol header) is seven bytes long. It consists of the following fields.



- Transaction Identifier - It is used for transaction pairing, the Modbus server copies in the response the transaction identifier of the request. (2 bytes)
- Protocol Identifier - It is used for intra- system multiplexing. The Modbus protocol is identified by the value 0. (2 bytes)
- Length - The length field is a byte count of the following fields, including the Unit Identifier and data fields. (2 bytes)
- Unit Identifier - This field is used for intra- system routing purpose. It is typically used to communicate to a Modbus or a Modbus+ serial line slave through a gateway between an Ethernet TCP/IP network and a Modbus serial line. This field is set by the Modbus Client in the request and must be returned with the same value in the response by the server. (1 byte)

This header provides some differences compared to the Modbus RTU application data unit used on serial line:

- The Modbus “slave address” field usually used on Modbus Serial Line is replaced by a single byte “Unit Identifier” within the MBAP Header. The “Unit Identifier” is used to communicate via devices such as bridges, routers and gateways that use a single IP address to support multiple independent Modbus end units.
- All Modbus requests and responses are designed in such a way that the recipient can verify that a message is finished. For function codes where the Modbus PDU has a fixed length, the function code alone is sufficient. For function codes carrying a variable amount of data in the request or response, the data field includes a byte count.
- Protocol Identifier - It is used for intra- system multiplexing. The Modbus protocol is identified by the value 0. (2 bytes)

The **function code field** of a message contains 8 bits. Valid function codes are in the range of 1-255 decimal. The function code instructs the slave what kind of action to take. Some examples are to read the status of a group of discrete inputs; to read the data in a group of registers; to write to an output coil or a group of registers; or to read the diagnostic status of a slave.

When a slave responds to the master, it uses the function code field to indicate either a normal response or that some type of error has occurred. For a normal response, the slave echoes the original function code. In an error condition, the slave echoes the original function code with its MSB set to a logic 1.

The **data field** is constructed using sets of two hexadecimal digits in the range of 00 to FF. According to the network's serial transmission mode, these digits can be made of a pair of ASCII characters or from one RTU character.

The data field also contains additional information that the slave uses to execute the action defined by the function code. This can include internal addresses, quantity of items to be handled, etc.

The data field of a response from a slave to a master contains the data requested if no error occurs. If an error occurs, the field contains an exception code that the master uses to determine the next action to be taken. The data field can be nonexistent in certain types of messages.



**NOTE:** ModScan32 is a Windows based application program that can be used as a Modbus master to access and change data points in a connected device (H0/H2/H4- ECOM100) The utility is ideally suited for quick and easy testing of Modbus TCP network slave devices. Visit [www.win-tech.com](http://www.win-tech.com) to download a free ModScan32 trial demo and for more information on ModScan32.

## Supported Modbus Function Codes

The following Modbus function codes are supported by the H0/H2/H4- ECOM100. Not all function codes are supported when the ECOM100 serves as a network client. The “Network Client Operation” section later in this chapter lists the function codes that are supported in client mode.

Modbus Function Code	Function	Server Mode	Client Mode
01	Read Discrete Output Table	Yes	Yes
02	Read Discrete Input Table		
03	Read Holding registers (when addressing mode is 584/984, this function is used to access analog output registers)		
04	Read Input Registers (when addressing mode is 584/984, this function is used to access analog input registers)		
05	Force Single Discrete Output		
06	Preset Single Holding Registers		
08	Loop Back / Maintenance		
15	Force Multiple Discrete Outputs		
16	Preset Multiple Holding Registers		
			Yes

## Network Server (Slave) Operation

This section describes how other Modbus TCP clients on a network can communicate with an H0/H2/H4 -ECOM100 that you have configured for Modbus TCP protocol. A network client must send a Modbus function code and Modbus address to specify a PLC memory location the DL05/06/205/405 CPU. No CPU ladder logic is required to support Modbus TCP server operation.

### Modbus Function Codes Supported

The H0/H2/H4 -ECOM100 supports the following Modbus function codes when acting as a Modbus TCP server.

Modbus Function Code	Function	DL05/06/205/405 Data Types Available
01	Read Discrete Output Table	Y, C, T, CT
02	Read Discrete Input Table	X, SP
03	Read Holding registers (when addressing mode is 584/984, this function is used to access analog output registers)	V
04	Read Input Registers (when addressing mode is 584/984, this function is used to access analog input registers)	V
05	Force Single Discrete Output	Y, C, T, CT
06	Preset Single Holding Registers	V
08	Loop Back / Maintenance	
15	Force Multiple Discrete Outputs	Y, C, T, CT
16	Preset Multiple Holding Registers	V

### Determining the Modbus Address

There are typically two ways that most Modbus addressing conventions allow you to specify a PLC memory location. These are:

- By specifying the Modbus data type and address
- By specifying a Modbus address only.

### If Your Host Software or Client Requires the Data Type and Address

Many Modbus TCP clients allow you to specify the Modbus data type and the Modbus address that corresponds to the PLC memory location. This is the easiest method, but not all packages allow you to do it this way.

The actual equation used to calculate the address depends on the type of PLC data you are using. The PLC memory types are split into two categories for this purpose.

- Discrete – X, SP, Y, C, S, T(contacts), CT (contacts)
- Word – V-Memory, Timer current value, Counter current value

In either case, you basically convert the PLC octal address to decimal and add the appropriate Modbus starting address (as required). The following tables show the exact range used for each group of data.



**NOTE:** For an automated Modbus/Koyo address conversion utility, download the file **Modbus\_conversion.xls** from the [www.automationdirect.com](http://www.automationdirect.com) technical support website.

DL05 Memory Type	Qty (Dec.)	PLC Range (Octal)	Modbus Address Range	Modbus Data Type
<b>For Discrete Data Types...</b>		<b>Convert PLC Addr. to Dec.</b>	<b>+ Start of Range</b>	<b>+ Data Type</b>
<b>Inputs (X)</b>	256	X0 – X377	2048 – 2303	Input
<b>Special Relays (SP)</b>	512	SP0 – SP777	3072 – 3583	Input
<b>Outputs (Y)</b>	256	Y0 – Y377	2048 – 2303	Coil
<b>Control Relays (C)</b>	512	C0 – C777	3072 – 3583	Coil
<b>Timer Contacts (T)</b>	128	T0 – T177	6144 – 6271	Coil
<b>Counter Contacts (CT)</b>	128	CT0 – CT177	6400 – 6527	Coil
<b>Stage Status Bits (S)</b>	256	S0 – S377	5120 – 5375	Coil
<b>For Word Data Types ...</b>		<b>Convert PLC Addr. to Dec.</b>	<b>+</b>	<b>Data Type</b>
<b>Timer Current Values (V)</b>	128	V0 – V177	0 – 127	Input Register
<b>Counter Current Values (V)</b>	128	V1000 – V1177	512 – 639	Input Register
<b>V-Memory, user data (V)</b>	3072	V1400 – V7377	768 – 3839	Holding Register

DL06 Memory Type	Qty (Dec.)	PLC Range (Octal)	Modbus Address Range	Modbus Data Type
For Discrete Data Types...		Convert PLC Addr. to Dec.	+ Start of Range	+ Data Type
Inputs (X)	512	X0 - X777	2048 - 2559	Input
Special Relays (SP)	512	SP0 - SP777	3072 - 3583	Input
Outputs (Y)	512	Y0 - Y777	2048 - 2559	Coil
Control Relays (C)	1024	C0 - C1777	3072 - 4095	Coil
Timer Contacts (T)	256	T0 - T377	6144 - 6399	Coil
Counter Contacts (CT)	128	CT0 - CT177	6400 - 6527	Coil
Stage Status Bits (S)	1024	S0 - S1777	5120 - 6143	Coil
Global Inputs (GX)	2048	GX0 - GX3777	0 - 2047	Input
Global Outputs (GY)	2048	GY0 - GY3777	0 - 2047	Coil
For Word Data Types ...		Convert PLC Addr. to Dec.	+	Data Type
Timer Current Values (V)	128	V0 - V177	0 - 127	Input Register
Counter Current Values (V)	128	V1000 - V1177	512 - 639	Input Register
V-Memory, user data (V)	256	V400 - V677	256 - 511	Holding Register
	3072	V1400 - V7377	768 - 3839	
	4096	V10000 - V17777	4096 - 8191	

D2-240 Memory Type	Qty (Dec.)	PLC Range (Octal)	Modbus Address Range	Modbus Data Type
For Discrete Data Types...		Convert PLC Addr. to Dec.	+ Start of Range	+ Data Type
Inputs (X)	320	X0 - X477	2048 - 2559	Input
Special Relays (SP)	144	SP0 - SP137 SP540 - SP617	3072 - 3167 3280 - 3471	Input
Outputs (Y)	320	Y0 - Y477	2048 - 2367	Coil
Control Relays (C)	256	C0 - C377	3072 - 3551	Coil
Timer Contacts (T)	128	T0 - T177	6144 - 6271	Coil
Counter Contacts (CT)	128	CT0 - CT177	6400 - 6527	Coil
Stage Status Bits (S)	512	S0 - S777	5120 - 5631	Coil
For Word Data Types ...		Convert PLC Addr. to Dec.	+	Data Type
Timer Current Values (V)	128	V0 - V177	0 - 127	Input Register
Counter Current Values (V)	128	V1000 - V1177	512 - 639	Input Register
V-Memory, user data (V)	1024	V2000 - V3777	1024 - 2047	Holding Register
V-Memory, user data (V) non-volatile	256	V4000 - V4377	2048 - 2303	Holding Register
V-Memory, system (V)	106	V7620 - V7737 V7746 - V7777	V3984 - V4063 V4070 - V4095	Holding Register

D2-250-1 Memory Type	Qty (Dec.)	PLC Range (Octal)	Modbus Address Range	Modbus Data Type
For Discrete Data Types...		Convert PLC Addr. to Dec.	+ Start of Range	+ Data Type
Inputs (X)	512	X0 – X777	2048 – 2560	Input
Special Relays (SP)	512	SP0 – SP137 SP320 – SP777	3072 – 3167 3280 – 3583	Input
Outputs (Y)	512	Y0 – Y777	2048 – 2560	Coil
Control Relays (C)	1024	C0 – C1777	3072 – 4095	Coil
Timer Contacts (T)	256	T0 – T377	6144 – 6399	Coil
Counter Contacts (CT)	128	CT0 – CT177	6400 – 6527	Coil
Stage Status Bits (S)	1024	S0 – S1777	5120 – 6143	Coil
For Word Data Types ...		Convert PLC Addr. to Dec.	+	Data Type
Timer Current Values (V)	256	V0 – V377	0 – 255	Input Register
Counter Current Values (V)	128	V1000 – V1177	512 – 639	Input Register
V-Memory, user data (V)	3072	V1400 – V7377	768 – 3839	Holding Register
	4096	V10000 – V17777	4096 – 8191	
V-Memory, system (V)	256	V7400 – V7777	3840 – 4095	Holding Register

D2-260/D2-262 Memory Type	Qty (Dec.)	PLC Range (Octal)	Modbus Address Range	Modbus Data Type
For Discrete Data Types...		Convert PLC Addr. to Dec.	+ Start of Range	+ Data Type
Inputs (X)	1024	X0 – X777	2048 – 3071	Input
Special Relays (SP)	512	SP0 – SP137 SP320 – SP717	3072 – 3167 3280 – 3535	Input
Outputs (Y)	1024	Y0 – Y1777	2048 – 3071	Coil
Control Relays (C)	2048	C0 – C3777	3072 – 5119	Coil
Timer Contacts (T)	256	T0 – T377	6144 – 6399	Coil
Counter Contacts (CT)	256	CT0 – CT377	6400 – 6655	Coil
Stage Status Bits (S)	1024	S0 – S1777	5120 – 6143	Coil
Global Inputs (GX)	2048	GX0 – GX3777	0 – 2047	Input
Global Outputs (GY)	2048	GY0 – GY3777	0 – 2047	Coil
For Word Data Types ...		Convert PLC Addr. to Dec.	+	Data Type
Timer Current Values (V)	256	V0 – V377	0 – 255	Input Register
Counter Current Values (V)	256	V1000 – V1377	512 – 639	Input Register
V-Memory, user data (V)	256	V400 – V777	256 – 511	Holding Register
	3072	V1400 – V7377	768 – 3839	
	11264	V10000 – V35777	4096 – 15359	
V-Memory, system (V)	256	V7600 – V7777 V36000 – V37777	3968 – 4095 15360 – 16383	Holding Register



D4-430 Memory Type	Qty (Dec.)	PLC Range (Octal)	Modbus Address Range (Decimal)	Modbus Data Type
For Discrete Data Types...		Convert PLC Addr. to Dec.	+ Start of Range	+ Data Type
Inputs (X)	320	X0 - X477	2048 - 3071	Input
Special Relays (SP)	288	SP0 - SP137 SP320 - SP617	3072 - 3167 3280 - 3471	Input
Outputs (Y)	320	Y0 - Y477	2048 - 2367	Coil
Control Relays (CR)	512	C0 - C737	3072 - 3583	Coil
Timer Contacts (T)	128	T0 - T177	6144 - 6271	Coil
Counter Contacts (CT)	128	CT0 - CT177	6400 - 6527	Coil
Stage Status Bits (S)	384	S0 - S577	5120 - 5503	Coil
Global I/O (GX)	512	GX0 - GX777	0 - 2047	Input
For Word Data Types ...		Convert PLC Addr. to Dec.	+	Data Type
Timer Current Values (V)	128	V0 - V377	0 - 255	Input Register
Counter Current Values (V)	128	V1000 - V1377	512 - 639	Input Register
V-Memory, user data (V)	3072	V1400 - V7377	768 - 3839	Holding Register
V-Memory, system (V)	256	V7400 - V7777	3840 - 4095	Holding Register

D4-440 Memory Type	Qty (Dec.)	PLC Range (Octal)	Modbus Address Range (Decimal)	Modbus Data Type
For Discrete Data Types...		Convert PLC Addr. to Dec.	+ Start of Range	+ Data Type
Inputs (X)	320	X0 - X477	2048 - 2367	Input
Special Relays (SP)	352	SP0 - SP137 SP320 - SP717	3072 - 3167 3280 - 3535	Input
Outputs (Y)	320	Y0 - Y477	2048 - 2367	Coil
Control Relays (CR)	1024	C0 - C1777	3072 - 4095	Coil
Timer Contacts (T)	256	T0 - T377	6144 - 6399	Coil
Counter Contacts (CT)	128	CT0 - CT177	6400 - 6527	Coil
Stage Status Bits (S)	1024	S0 - S1777	5120 - 6143	Coil
Global I/O (GX)	1024	GX0 - GX1777	0 - 1023	Input
For Word Data Types ...		Convert PLC Addr. to Dec.	+	Data Type
Timer Current Values (V)	256	V0 - V377	0 - 255	Input Register
Counter Current Values (V)	128	V1000 - V1377	512 - 639	Input Register
V-Memory, user data (V)	3072 4096	V1400 - V7377 V10000 - V17777	768 - 3839 4096 - 8191	Holding Register
V-Memory, system (V)	268	V700 - V737 V7400 - V7777	448 - 479 3840 - 4095	Holding Register

D4-450/D4-454 Memory Type	Qty (Dec.)	PLC Range (Octal)	Modbus Address Range	Modbus Data Type
For Discrete Data Types...		Convert PLC Addr. to Dec.	+ Start of Range	+ Data Type
Inputs (X)	1024	X0 – X777	2048 – 3071	Input
Special Relays (SP)	512	SP0 – SP137 SP320 – SP717	3072 – 3167 3280 – 3535	Input
Outputs (Y)	1024	Y0 – Y1777	2048 – 3071	Coil
Control Relays (C)	2048	C0 – C3777	3072 – 5119	Coil
Timer Contacts (T)	256	T0 – T377	6144 – 6399	Coil
Counter Contacts (CT)	256	CT0 – CT377	6400 – 6655	Coil
Stage Status Bits (S)	1024	S0 – S1777	5120 – 6143	Coil
Global Inputs (GX)	1536	GX0 – GX2777	0 – 1535	Input
Global Outputs (GY)	1536	GY0 – GY2777	0 – 1535	Coil
For Word Data Types ...		Convert PLC Addr. to Dec.	+	Data Type
Timer Current Values (V)	256	V0 – V377	0 – 255	Input Register
Counter Current Values (V)	256	V1000 – V1377	512 – 767	Input Register
V-Memory, user data (V)	3072 12288	V1400 – V7377 V10000 – V37777	768 – 3839 4096 – 16383	Holding Register
V-Memory, system (V)	320	V700 – V777 V7400 – V7777	448 – 768 3968 – 4095	Holding Register

The following examples show how to generate the Modbus address and data type for hosts which require this format.

### Example 1: V2100

Find the Modbus address for User V location V2100.

1. Find V- Memory in the table.
2. Convert V2100 into decimal (1088).
3. Use the Modbus data type from the table.

PLC Addr. (Dec.) + Data Type

V2100 = 1088 decimal

1088 + Hold. Reg. = Holding Reg.

Timer Current Values (V)	128	V0-V177	0-127	Input Register
Counter Current Values (V)	128	V1000-V1177	512-639	Input Register
V-Memory, user data (V)	1024	V2000-V3777	1024-2047	Holding Register

### Example 2: Y20

Find the Modbus address for output Y20.

1. Find Y outputs in the table.
2. Convert Y20 into decimal (16).
3. Add the starting address for the range (2049).
4. Use the Modbus data type from the table.

PLC Addr.(Dec.) + Start Addr. + Data Type

Y20 = 16 decimal

16 + 2049 + Coil = Coil 2065

Outputs (Y)	320	Y0-Y477	2049-2367	Coil
Control Relays (C)	256	C0-C377	3072-3551	Coil

### Example 3: T10 Current Value

Find the Modbus address to obtain the current value from Timer T10.

1. Find Timer Current Values in the table.
2. Convert T10 into decimal (8).
3. Use the Modbus data type from the table.

PLC Addr.(Dec.) + Data Type

T10 = 8 decimal

8 + Input Reg. = Input Reg. 8

Timer Current Values (V)	128	V0-V177	0-127	Input Register
Counter Current Values (V)	128	V1000-V1177	512-639	Input Register

**Example 4: C54**

Find the Modbus address for Control Relay C54

1. Find Control Relays in the table.
2. Convert C54 into decimal (44).
3. Add the starting address for the range (3072).
4. Use the Modbus data type from the table.

PLC Addr.(Dec.) + Start Addr. + Data

C54= 44 decimal

$$44 + 3072 + \text{Coil} = \text{Coil 3117}$$

Outputs (Y)	320	Y0-Y477	2049-2367	Coil
Control Relays (CR)	256	C0-C377	3072-3551	Coil

**If the Host Software or Client Requires an Address ONLY**

Some Modbus TCP clients do not allow you to specify the Modbus data type and address. Instead, you specify an address only. This method requires another step to determine the address, but it is not difficult. Basically, Modbus also separates the data types by address ranges as well. This means an address alone can actually describe the type of data and location. This is often referred to as “adding the offset”.

The actual equation used to calculate the address depends on the type of PLC data you are using. The PLC memory types are split into two categories for this purpose.

- Discrete – X, GX, SP, Y, CR, S, T, C (contacts)
- Word – V-memory , Timer current value, Counter current value

In either case, you basically convert the PLC octal address to decimal and add the appropriate Modbus starting address (as required). The following tables show the exact range used for each group of data.



**NOTE:** For an automated Modbus/Koyo address conversion utility, download the file *Modbus\_conversion.xls* from the [www.automationdirect.com](http://www.automationdirect.com) website.

Discrete Data Types*				
PLC Memory Type	Qty (Dec.)	PLC Range (Octal)	Modbus Address Range	Access
Global Inputs (GX)	2048	GX0-GX1746	0-1023	Read Only
Inputs (X)	1024	X0-X1777	2048-2367	
Special Relays (SP)	512	SP0-SP777	13073-13584	
Reserved	-	-	13585-20000	
Global Outputs (GY)	2048	GY0-GY3777	0-2048	Read/Write
Outputs (Y)	1024	Y0-Y1777	2049-3072	
Control Relays (CR)	2048	C0-C3777	3073-5012	
Timer Contacts (T)	256	T0-T377	6145-6400	
Counter Contacts (CT)	256	CT0-CT377	6401-6656	
Stage Status Bits (S)	1024	S0-S1777	5121-6144	
Reserved	-	-	6657-10000	

\* Refer to your PLC user manual for the correct memory mapping size of your PLC. Some of the addresses shown above might not pertain to your particular CPU.

Word Data Types*					
Registers (Word) (V-Memory)	Qty (Dec.)	PLC Range (Octal)	Modbus 40001 Address Range	Modbus 30001 Address Range	Access
<b>Timers</b>	256	V0 - V377	40001 - 40256	30001 - 30256	Read/Write
<b>Counters</b>	256	V1000 - V1377	40513 - 40768	30513 - 30768	
<b>Data Words</b>	256	V400 - V777	40257 - 40512	30257 - 30512	
	3072	V1400 - V7377	40769 - 43840	30769 - 33840	
	5906	V10000 - V23416	44097 - 49999	34097 - 39999	
	5361	V23417 - V35777	410000 - 415360	310000 - 315360	
<b>System Parameters</b>	128	V7600 - V7777	43969 - 44096	33969 - 34096	
	1024	V36000 - V37777	415361 - 416384	315361 - 316384	Read Only
<b>Remote Inputs</b>	128	V40000 - V40177	416385 - 416512	316385 - 316512	
<b>Remote Outputs</b>	128	V40200 - V40377	416513 - 416640	316513 - 316640	Read/Write
<b>Input Points</b>	64	V40400 - V40477	416641 - 416704	316641 - 316704	Read Only
<b>Output Points</b>	64	V40500 - V40577	416705 - 416768	316705 - 316768	Read/Write
<b>Control Relays</b>	128	V40600 - V40777	416769 - 416896	316769 - 316896	
<b>Timer Status Bits</b>	16	V41100 - V41117	416961 - 416976	316961 - 316976	
<b>Counter Status Bits</b>	16	V41140 - V41157	416993 - 417008	316993 - 317008	Read Only
<b>Special Relays</b>	32	V41200 - V41237	417025 - 417056	317025 - 317056	

\* Refer to your PLC user manual for the correct memory mapping size of your PLC. Some of the addresses shown above might not pertain to your particular CPU.

The following examples show how to generate the Modbus address and data type for hosts which require this format.

**Example 1: V2100**

Find the Modbus address for User V location V2100.

1. Find V-memory in the table.
2. Convert V2100 into decimal (1088).
3. Add the Modbus starting address for the mode (40001).

**PLC Addr. (Dec.) + Mode Address**

V2100 = 1088 decimal

$$1088 + 40001 = 41089$$

For Word Data Types ...	PLC Address Dec.	+	Appropriate Mode Address
Timer Current Values (V)	128	V0-V177	0-127
Counter Current Values (V)	128	V1000-V1177	512-639
V-Memory, user data (V)	1024	V2000-V3777	1024-2047
			3001 30001 Input Reg.
			3001 30001 Input Reg.
			4001 40001 Holding Reg.

**Example 2: Y20**

Find the Modbus address for output Y20.

1. Find Y outputs in the table.
2. Convert Y20 into decimal (16).
3. Add the starting address for the range (2048).
4. Add the Modbus address for the mode (1).

**PLC Addr.(Dec.) + Start Address + Mode**

Y20 = 16 decimal

$$16 + 2048 + 1 = 2065$$

Outputs (Y)	320	Y0-Y477	2048-2367	1	1	Coil
Control Relays (CR)	256	C0-C377	3072-3551	1	1	Coil
Timer Contacts (T)	128	T0-T177	6144-6271	1	1	Coil

**Example 3: C54**

Find the Modbus address for Control Relay C54.

1. Find Control Relays in the table.
2. Convert C54 into decimal (44).
3. Add the starting address for the range (3072).
4. Add the Modbus address for the mode (1).

**PLC Addr.(Dec.) + Start Address + Mode**

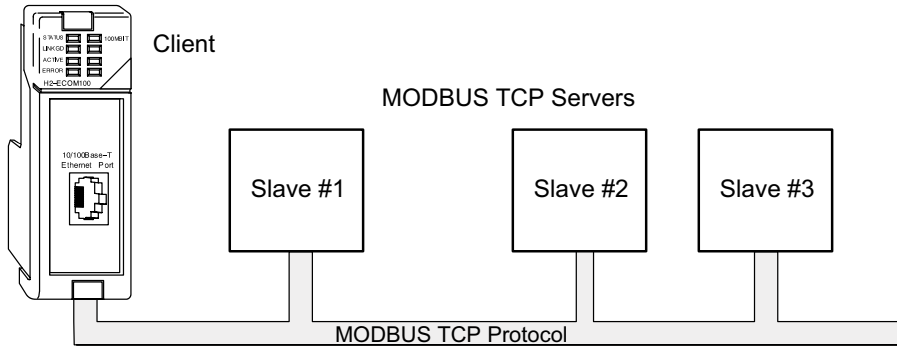
C54 = 44 decimal

$$44 + 3072 + 1 = 3117$$

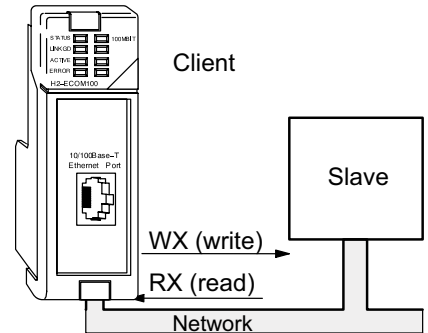
Outputs (Y)	320	Y0-Y477	2048-2367	1	1	Coil
Control Relays (CR)	256	C0-C377	3072-3551	1	1	Coil
Timer Contacts (T)	128	T0-T177	6144-6271	1	1	Coil

## Network Client (Master) Operation

This section describes how the DL05/06/205/405 CPU can serve as a client on a Modbus TCP network using the H0/H2/H4-ECOM100. This section discusses how to design the required ladder logic for network client operation.



When using the ECOM100 as a client on the network, you use simple RLL instructions to initiate the requests. The WX instruction initiates network write operations, and the RX instruction initiates network read operations. Before executing either the WX or RX commands, we need to load data related to the read or write operation onto the CPU's accumulator stack. When the WX or RX instruction executes, it uses the information on the stack combined with data in the instruction box to completely define the task.



## Modbus Function Codes Supported

The H0/H2/H4 -ECOM100 supports the following Modbus function codes when acting as a Modbus TCP client.

ECOM100 as Modbus TCP Client					
PLC Instruction	Operand	Modbus		FC	Description
		Base	Offset		
RX	GY0 – GY3777	00000	1 – 2048	01	Read Coils
	Y0 – Y1777		2049 – 3072		
	C0 – C3777		3073 – 5120		
	S0 – S1777		5121 – 6144		
	T0 – T377		6144 – 6400		
	CT0 – CT377		6401 – 6656		
	GX0 – GX3777	10000	1 – 2048	02	Read Inputs
	X0 – X1777		2049 – 3072		
	SP0 – SP777		3073 – 3584		
	V0 – V41117	40000	1 – 16976	03	Read Holding Registers
V41140 – V41157	16993 – 17008				
V41200 – V41237	17025 – 17056				
WX	GY0 – GY3777	00000	1 – 2048	15	Force Multiple Coils
	Y0 – Y1777		2049 – 3072		
	C0 – C3777		3073 – 5120		
	S0 – S1777		5121 – 6144		
	T0 – T377		6145 – 6400		
	CT0 – CT377		6401 – 6656		
	GX0 – GX3777	10000	1 – 2048	-	-
	X0 – X1777		2049 – 3072		
	SP0 – SP777		3073 – 3584		
	V0 – V41117	40000	1 – 16976	16	Preset Multiple Registers
	V41140 – V41157		16993 – 17008		
	V41200 – V41237		17025 – 17056		
	V0 – V41117 (odd length)	30000	1 – 16976	-	-
V41140 – V41157 (odd length)	16993 – 17008				
V41200 – V41237 (odd length)	17025 – 17056				

**NOTE:** The H0/H2/H4- ECOM100, as a client/master, supports function code 4. Thus, 30001 address ranges can be read from a server/slave device.

This is done by specifying an odd number of bytes transferred instead of the normal even number of bytes. Thus: Even number of bytes to transfer: RX/WX for the Holding Registers (400001+ address range).

Odd number of bytes to transfer: RX for the Input Registers (30001+ address range). It is not possible to use WX on 30001 address ranges because by definition Input Registers are "read-only."





## PLC Memory Supported for Client Operation

The actual equation used to calculate the address depends on the type of PLC data you are using. The PLC memory types are split into three categories for this purpose.

- Discrete Inputs - GX, X, SP
- Discrete Outputs - GY, Y, CR, T, CT, S
- Word - Timer current value, Counter current value, Data Words

In either case, you basically take the Modbus address you are trying to target, subtract the starting Modbus of that range, convert the result to octal and add the octal number to the beginning PLC address in the appropriate PLC range. See the conversion examples on the following page. The following tables show the exact range used for each group of data.



**NOTE:** For an automated Modbus/Koyo address conversion utility, download the file *Modbus\_conversion.xls* from the [www.automationdirect.com](http://www.automationdirect.com) website.

Discrete Data Types*				
PLC Memory Type	Qty (Dec.)	PLC Range (Octal)	Modbus Address Range	Access
Global Inputs (GX)	2048	GX0 - GX1746	10001 - 10999	Read Only
Inputs (X)	1024	GX1747 - GX3777	11000 - 12048	
Special Relays (SP)	512	X0 - X1777	12049 - 13072	
Reserved	-	SP0 - SP777	13073 - 13584	
Global Outputs (GY)	2048	-	13585 - 20000	Read/Write
Outputs (Y)	2048	GY0 - GY3777	1 - 2048	
Control Relays (CR)	1024	Y0 - Y1777	2049 - 3072	
Timer Contacts (T)	2048	C0 - C3777	3073 - 5120	
Counter Contacts (CT)	256	T0 - T377	6145 - 6400	
Stage Status Bits (S)	256	CT0 - CT377	6401 - 6656	
Reserved	1024	S0 - S1777	5121 - 6144	
Reserved	-	-	6657 - 10000	

Word Data Types*				
Registers (Word) (V-Memory)	Qty (Dec.)	PLC Range (Octal)	Modbus Address Range	Access
Timers	256	V0 - V377	40001 - 40256	Read/Write
Counters	256	V1000 - V1377	40513 - 40768	
Data Words	256	V400 - V777	40257 - 40512	
	3072	V1400 - V7377	40769 - 43840	
	5903	V10000 - V23416	44097 - 49999	
	5361	V23417 - V35777	410000 - 415360	
System Parameters	128	V7600 - V7777	43969 - 44096	
	1024	V36000 - V37777	415361 - 416384	

\* Refer to your PLC user manual for the correct memory mapping size of your PLC. Some of the addresses shown above might not pertain to your particular CPU.



**NOTE:** Your PC's Windows calculator can be used for number conversions (i.e. decimal to octal). The Windows calculator must be in Calculator>View>Scientific mode to enable number conversions capability.

---

### Example 1: Calculating Word PLC Address

Find the PLC address to use to target Modbus address 41025 in a server device.

1. Subtract the beginning of the Modbus word address range (40001) from the desired Modbus address to target.  
1.  $41025 - 40001 = 1024$  decimal
2. Convert decimal result into octal.  
2.  $1024$  decimal = 2000 octal
3. Add octal result to beginning PLC range (Input, Output or Word).  
3.  $V0$  (octal) + 2000 (octal) = **V2000** octal

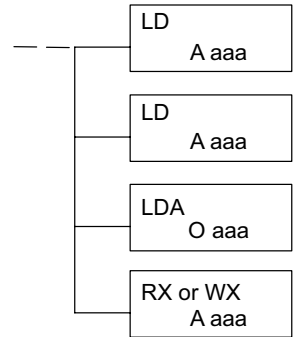
### Example 2: Calculating Discrete Input PLC Address

Find the PLC address to use to target Modbus address 12060 in a server device.

1. Subtract the beginning of the Modbus Input address range (12049) from the desired Modbus address to target.  
1.  $12060 - 12049 = 11$  decimal
2. Convert decimal result into octal.  
2.  $11$  decimal = 13 octal
3. Add octal result to beginning PLC range (Input, Output or Word).  
3.  $X0$  (octal) + 13 octal = **X13** octal

### Building the Read (RX) or Write (WX) Routine

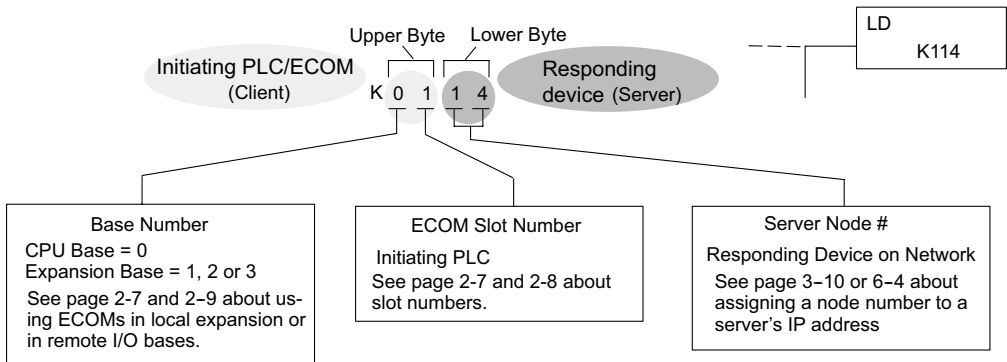
For network communications, you build the Read (RX) or Write (WX) instructions into a routine which requires the four instructions you see to the right. They must be used in the sequence shown. The following step-by-step procedure will provide you the information necessary to set up your ladder program to receive data from a network server.



**NOTE:** Please review intelligent instructions (IBOX) in Chapter 5 of the user manual for the PLC you are using, which simplifies this and other functions. Consider the following IBOX instructions: ECOM100, ECRX and ECWX.

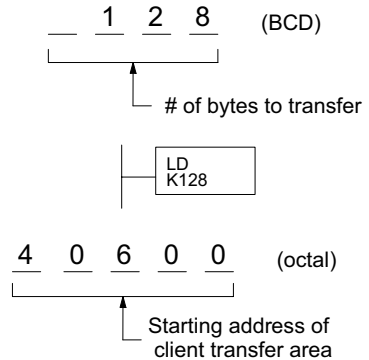
### Step 1: Identify ECOM Slot Location and Server Node #

The first Load (LD) instruction accepts either a constant or a variable. Use a “K” to designate the number as a constant. Use a “V” if you are entering the address of a register. The contents of that register perform the same function as the constant shown below. For example, you could use V2000 in place of K0114. If the contents of V2000 is the number “114,” the function would be the same. Using a variable allows changing parameters while the program is running.



### Step 2: Load Number of Bytes to Transfer

The second Load (LD) instruction determines the number of bytes which will be transferred between the master and slave in the subsequent WX or RX instruction. The value to be loaded is in BCD format (decimal), from 1 to 128 bytes. Requesting an even number of bytes, generates a Modbus message using Function 03, Read Holding Registers. If you need to Read Input Registers, Function Code 04, enter an odd number of bytes. For example, to read 10 Input Holding Registers, enter 2 (bytes/word) X 10 registers + 1, 21 bytes. This will request ten 30001 range addresses from the Modbus server (slave) device.

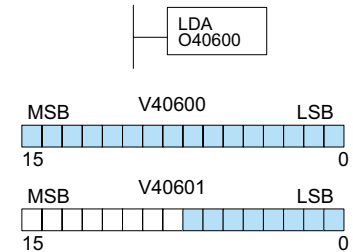


### Step 3: Specify Master Memory Area

The third instruction in the RX or WX sequence is a Load Address (LDA) instruction. Its purpose is to load the starting address of the memory area to be transferred. Entered as an octal number, the LDA instruction converts it to hex and places the result in the accumulator.

For a WX instruction, the CPU sends the number of bytes previously specified from its memory area beginning at the LDA address specified.

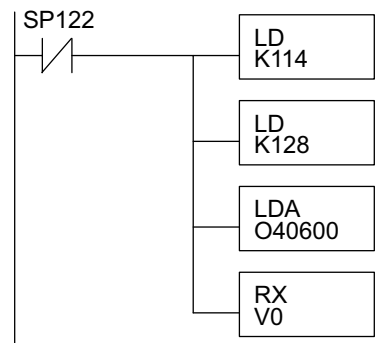
For an RX instruction, the CPU reads the number of bytes previously specified from the server, placing the received data into its memory area beginning at the LDA address specified.



**NOTE:** Since V-memory words are always 16 bits, you may not always use the whole word. For example, if you only specify to read 3 bytes, you will only get 24 bits of data. In this case, only the 8 least significant bits of the last word location will be modified. The remaining 8 bits are not affected.

### Step 4: Specify Slave Memory Area

The last instruction in our sequence is the WX or RX instruction itself. Use WX to write to the server, and RX to read from the server. All four of our instructions are shown to the right. In the last instruction, you must specify the starting address and a valid data type for the server.



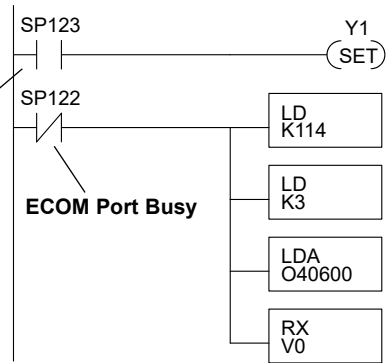
### Communications from a Ladder Program

Typically network communications will last longer than 1 scan. The program must wait for the communications to finish before starting the next transaction.

**ECOM Communication Error**

Depending on which slot the ECOM is in, it has two Special Relay contacts associated with it (see page 4-11 to 4-12 for special relays). One indicates “Port busy”, and the other indicates “Port Communication Error”. The example at right shows the use of these contacts for an ECOM that is in slot 1. The “Port Busy” bit is on while the PLC communicates with the slave. When the bit is off the program can initiate the next network request.

The “Port Communication Error” bit turns on when the PLC has detected an error. Use of this bit is optional. When used, it should be ahead of any network instruction boxes since it will be reset when an RX or WX instruction is executed.

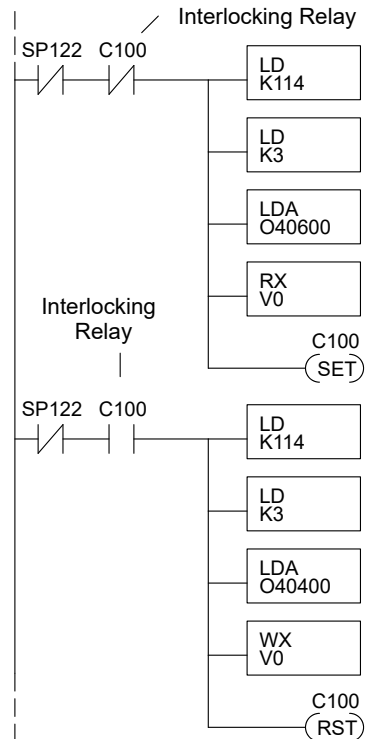


### Multiple Read and Write Interlocks

If you are using multiple reads and writes in the RLL program, you have to interlock the routines to make sure all the routines are executed. If you don't use the interlocks, then the CPU will only execute the first routine. This is because each port can only handle one transaction at a time.

In the example to the right, after the RX instruction is executed, C100 is set. When the port has finished the communication task, the second routine is executed and C100 is reset.

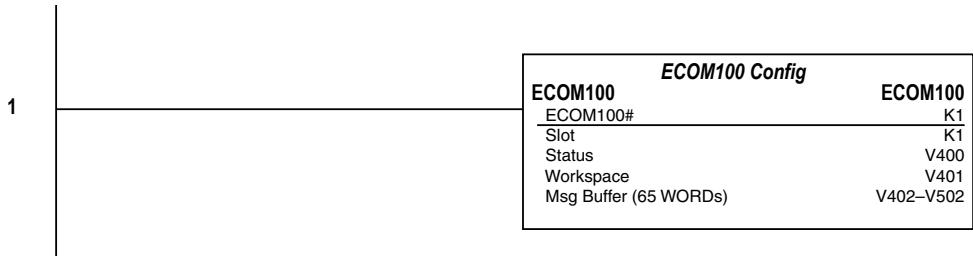
If you are using RLL<sup>PLUS</sup> Stage Programming, you can put each routine in a separate program stage to ensure proper execution and switch from stage to stage allowing only one of them to be active at a time.



## ECOM100 IBOX

The following information is an explanation of how to use IBox instructions when using ECOM100s for Modbus TCP. There are 2 specific IBOX's that can help with Modbus TCP communications (ECRX and ECWX) and another IBOX (ECOM100) that must be used to sequence these instructions. Use this information in conjunction with the material covered earlier in this chapter on Modbus functionality.

The ECOM100 IBOX must be placed at the top of ladder, with no input logic. You will need one box for each ECOM100 you wish to use. The slot location of the ECOM100 is assigned to an ECOM # here, as well as the address ranges needed by the instruction. This range MUST



be unique and cannot be used for any other purpose. The same is true for ANY workspace V-memory assignment in any IBOX used.

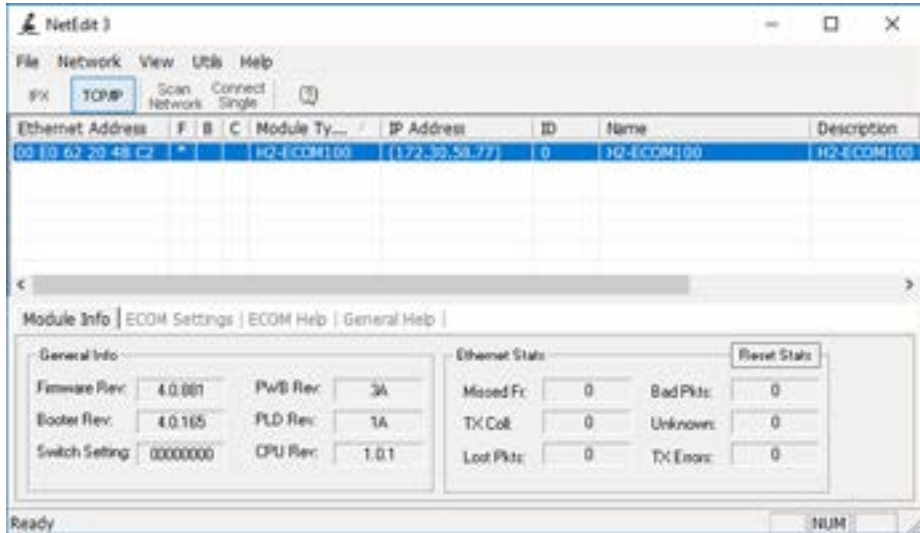
These instructions auto-sequence themselves, but *Direct*Logic octal addressing still must be used. There is a Modbus spreadsheet located on our Tech Support site that can be used to convert the Modbus addresses in the slaves to octal addressing that is required in the ECRX and ECWX boxes.

Follow this link: [http://support.automationdirect.com/docs/modbus\\_conversion.xls](http://support.automationdirect.com/docs/modbus_conversion.xls), or use this application note AN-MISC-010, which is located <https://support.automationdirect.com/technotes.html> and select AN-MISC-010 to download the spreadsheet.

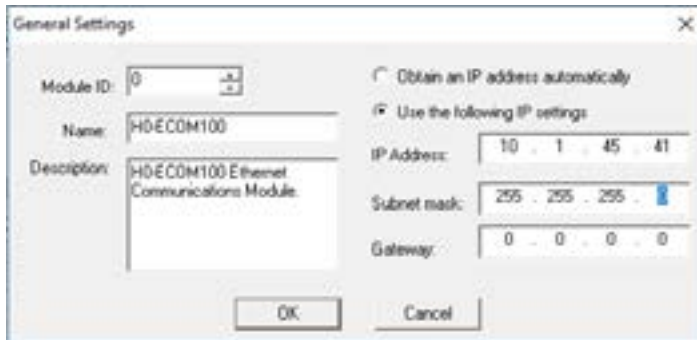
Once you have used the spreadsheet to determine the appropriate DL address to use for your desired Modbus address, place this value into the “From Slave Element” field. Enter the # of bytes you wish to retrieve, and enter the appropriate DL address you want to receive this data. The “Slave ID” at this time has no bearing to the actual slave, it will be tied to the IP address of the Modbus device in the NetEdit and Peer-Peer setup below.

For example, if you want to read the first Modbus coils, you would place “GY1” in “From Slave Address”, the # of bytes, and you could place the data into C400, for example. For Holding Registers, you would place the data into V-memory locations. The example program at the bottom will READ V40001 and V40002 from the Modbus slave, and immediately WRITE V7766 and V7767 ( RTC Seconds and Minutes) to 40003 and 40004. So if the slave has no data there, the PLC will send Seconds and Minutes values (except the DL05 which will send 0 unless it has the Real-Time Clock module installed).

For the final steps, you must setup the ECOM100 IP settings and “Peer-Peer” table with NetEdit3.

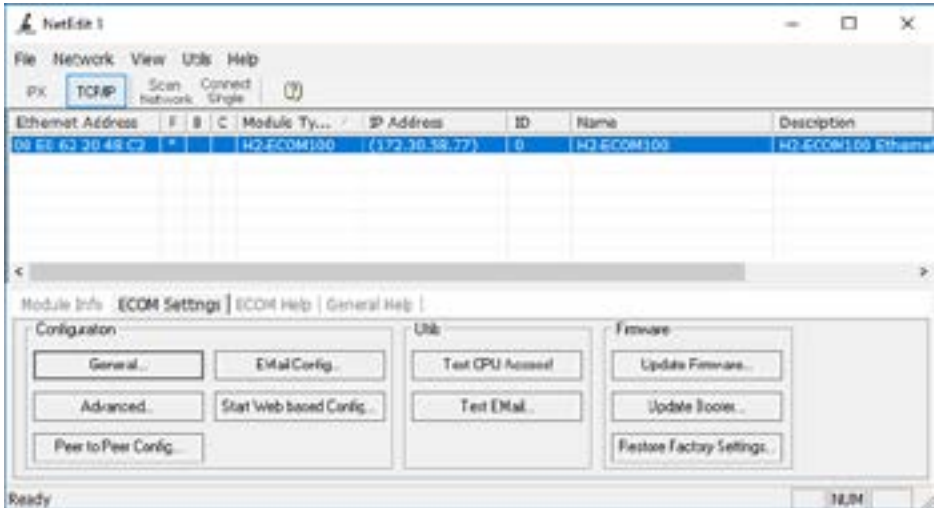


In the graphic above, double-click the ECOM100 desired. This will pull up the following **General Settings** box. Here, you can assign the IP address and subnet mask for his network. The Modbus slaves will need to have compatible settings of course. When finished , click OK to go back to NetEdit main screen.

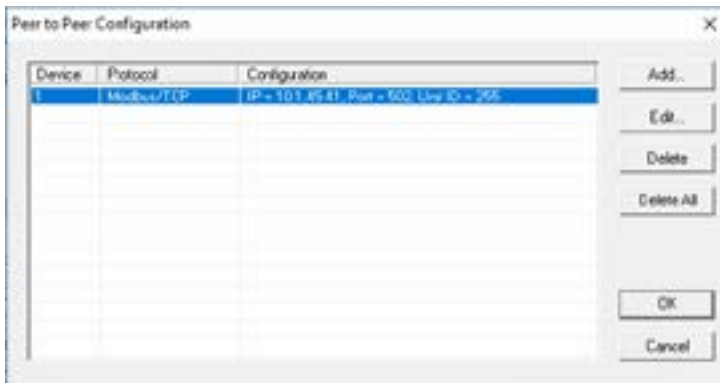


### Modbus TCP Setup

To begin the Modbus TCP setup, highlight the desired ECOM100 module, and select the “ECOM Settings” tab in lower frame of NetEdit3 window, then click the button marked “Peer to Peer Config”.



You will get the popup screen seen below:



To add an entry, click the “Add” button, or “Edit” button to make changes to the existing settings. Here we will choose “Add”. The **Add Device Address** popup will open.



Make sure to check the box for “ModbusTCP” and make “Device #” equal to the “Slave ID” that was used in the ECxX box(s), and enter the IP address of the Modbus slave. Leave the UnitID at 255 and port at 502 unless your slave documentation instructs otherwise.



**NOTE:** *UnitID is normally used with ethernet-serial gateways, where one IP address (the gateway) may be used for many serial slaves. Here, the UnitID would represent the serial slave address, and would need to be changed to match the various slaves.*

So the IP address might be 10.1.45.34 for a Modbus gateway and the UnitID would be “1” for the first serial slave, “2” for the second, and so on.

Once you click “OK”, the ECOM100 will be updated, there is no further action to take place with NetEdit until you are ready to add additional information to the table.

Device	Protocol	Configuration
1	Modbus/TCP	IP = 10.1.45.33, Port = 502, Unit ID = 255
2	Modbus/TCP	IP = 10.1.45.34, Port = 502, Unit ID = 255

### Example Modbus TCP Program

On the following pages is an example RLL program using Device ID 1 from the Peer to Peer Configuration box. The user could easily add additional IBOXs to implement other devices.

After creating your program, ensure your program has an END statement. Select “Accept” ( F8), and write the project to the PLC (Shift+F9). Make sure after you have written the project, that the PLC is placed into Program mode, then back into Run mode (many IBOXs are only processed after a Program-to-Run transition). If everything is correct, V505 ( RX/ WX OK Count) using the Success bit, should be incrementing very fast.

### Troubleshooting:

**Modbus Addressing:** You must know the addressing of your device. Some devices give addresses in hex values, which can often appear like a decimal value (310 is a valid hex or decimal value). Asian drives often use hex. You can use Windows calculator “scientific” view to convert the addresses.

Also, many devices use addresses as “offsets”, particularly Holding Registers. The address 40001 means the first Holding Register, but your device may term this as Holding Register 0 or 1 (the 40000 is assumed). Whether this is actually an address of 0 or 1 is hard to predict.

An excellent method of troubleshooting is to try to only read from the middle of a known address range, that will have non-zero values. If you know the device has 10 Holding registers starting at 1, try to read #3, and compare that to the values in the device. If that is one more or one less than you expect, then that is the offset you will have to use in the *DirectLogic* numeric conversion.

- Make sure you can ping your device
- Make sure the PLC has made a Program-to-Run transition
- Make sure the LinkGood light is ON, on the ECOM100 module
- Make sure Dipswitch 7 is ON, on the ECOM100 module

On rare occasions, and almost exclusively with Festo or Numatics devices, their Holding Registers start at a VERY high number, around 45,000. There isn't an equivalent octal address to convert that high a value, so a Z constant was introduced that allows the use of a hex value in the “From Slave Element” field. So a 45,392 address is actually the offset added to 40,000... $45,392 = B150$  hex, so the entry in the “From Slave Element” field would be ZB150.

There are 3rd party Modbus shareware programs available on the internet, and we have a free ModbusTCP tester at the link below. You might need to use one of these programs to test to your device, to make sure they can work successfully to the expected addressing.

**[http://ftp.automationdirect.com/pub/Modbus\\_TCP\\_Master.zip](http://ftp.automationdirect.com/pub/Modbus_TCP_Master.zip)**

To read Modbus Input Registers, you must change the number of bytes in any instruction to the next odd number. For example, if you are reading 16 bytes (8 Registers), then you would increase the number by 1 to 17 bytes, and the instruction will then be trying to read Modbus 30000 addresses.

Network #1 uses the Hx-ECOM100 in Slot 1. This would be the only slot in 05, first slot in 06, and second slot in 205/405 models. It will use the range of V-memory from V400 - V502 as the working status, workspace and buffer. These locations must not be used anywhere else.

Make sure Dipswitch 7 is turned ON in the ECOM100.

The Modbus converter spreadsheet from ADC Tech Support site will be extremely useful to convert Modbus addresses into the octal-based addressing required in the instructions.

NOTE: NetEdit 3 MUST be used to setup the ECOM100 "Peer-to-Peer Config" table in the ECOM100. This is what determines if the communications are ModbusTCP or ECOM.

<i>ECOM100 Config</i>	
ECOM100	ECOM100
ECOM100#	K1
Slot	K1
Status	V400
Workspace	V401
Msg Buffer (65 WORDs)	V402-V502

On the first PLC scan, set the Comm Success & Comm Error count registers to 0.

Also SETS C106, which is the enable logic to the ECxX boxes.



Once the ECRX and ECWX IBoxes are enabled, the ECOM100 IBox will automatically sequence them, no manual control of the port busy bits is required.

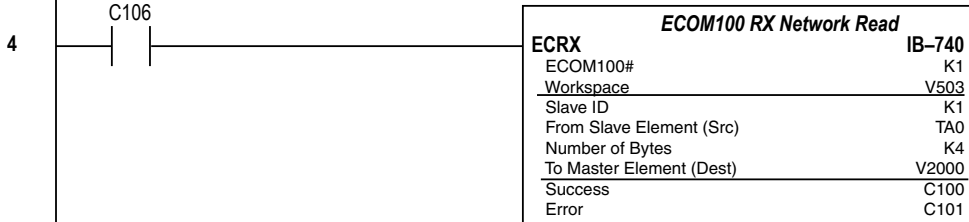
This example uses C106 with a SET on First Scan, it could be changed to whatever logic the user desires such as SP1.

3 \_\_\_\_\_ ( NOP )

The ECRX will read from Slave ID "1", and will target address TA0 (V0) which is the Modbus equivalent address 40001 ( first Holding Register).

It will get 4 bytes ( 2 registers) and place the data in V2000-2001. This data will likely be in decimal format if coming from 3rd party devices. The DataView window at left has V2000 -2001 set for Decimal format.

Note that Workspace V location must be unique.

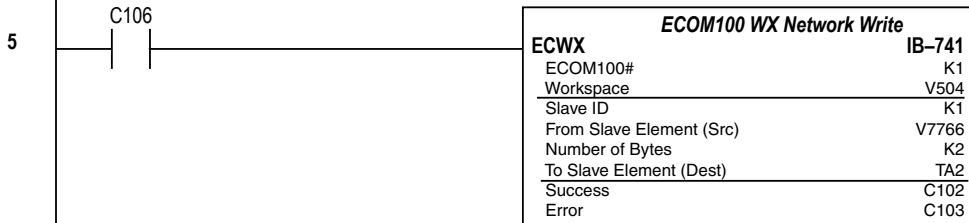


The ECWX will write to Slave ID "1", and will target address TA2 ( V2) which is the Modbus equivalent of 40003 ( third Holding Register).

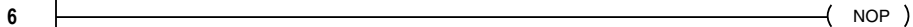
It will write 4 bytes (2 registers) from the PLC addresses V7766-V7767 . These 2 addresses are the Seconds and Minutes from the PLC RealTime Clock, so they are nonzero most of the time. Note the data will be in BCD/Hex format.

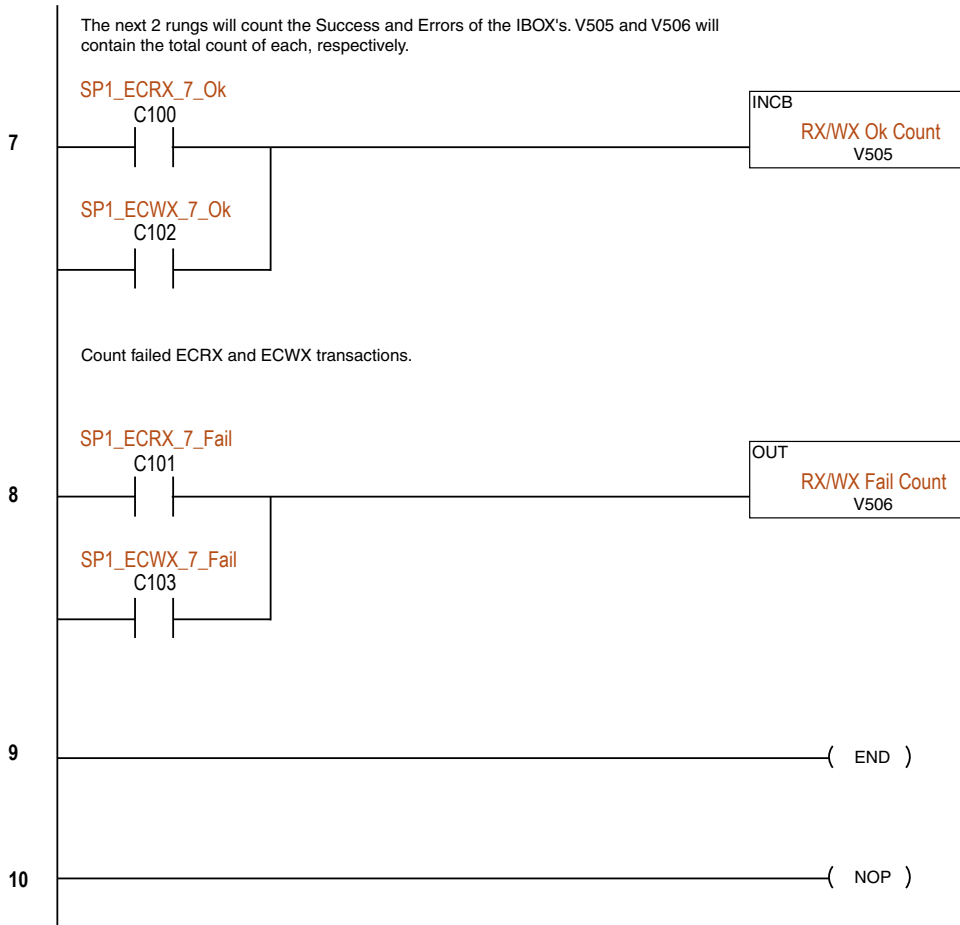
The DL05 will only have data here if using the Option module D0-01MC, otherwise it will be zeroes.

Note that Workspace location must be unique.



NOTE: If trying to read Modbus Input Registers ( Function Code 4 or 30001 addressing), the # of bytes must be increased by 1 to the next odd number. This is how the DirectLogic and ECOM100 recognize the Modbus address is an Input Register.





## H0/H2/H4 - ECOM100 System Memory

H0/H2/H4 - ECOM100				
	Modbus Address Range (Decimal)	Words (16-bit)	Word Descriptions	Access
Module Version Information	317501 – 317506 (417501 – 417506)*	6	1 - OS Major Version 2 - OS Minor Version 3 - OS Build Version 4 - Booter Major Version 5 - Booter Minor Version 6 - Booter Build Version	Read Only
	317507 – 317510 (417507 – 417510)	-	Reserved	-
Device Data	317511 – 317600 (417511 – 417600)*	90	1 - Version of Device 2 - Family 3 - Processor 4 - Module Type 5 - Status Code (6 – 8) - Ethernet Address 9 - RAM Size 10 - Flash Size 11 - Batt RAM Size 12 - DIP Settings 13 - Media Type (14 – 15) - EPF Count (if supported) 16 - Run Relay State (if supported) 17 - Batt Low (if supported) 18 - Model Number 19 - Ethernet Speed (20 – 90) - Reserved	Read Only
	317601 – 318500 (417601 – 418500)	-	Reserved	-
Dynamic Module Data	418001 – 418020	20	(1–3) -- Reserved 4 - Flags: Bit 0: If 1, module has rebooted since this bit was cleared, a write to the Flags word with this bit set will clear this reboot bit. Bit (1–7) -- Reserved 5 - Reboot Count (LSW) - Read Only 6 - Reboot Count (MSW) - Read Only (7–20) - Reserved	Read/Write
	418021 – 419250	-	Reserved	-

\*For clients that only support function code 3 to read word data.