MODBUS® TCP/IP for H0/H2/H4–ECOM100

5

In This Chapter. . . .

— MODBUS TCP/IP
— Supported MODBUS Function Codes
— Network Server Operation
— Network Client Operation
— H0/H2/H4–ECOM100 System Memory

MODBUS TCP/IP

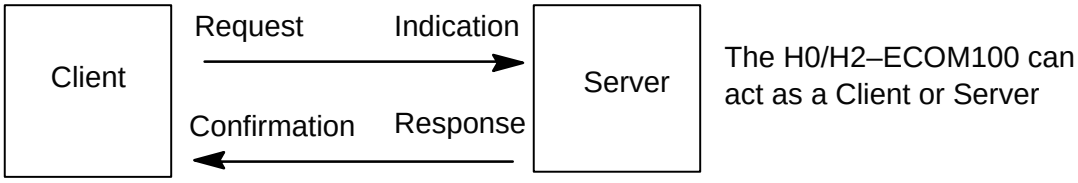
MODBUS TCP/IP is essentially the serial MODBUS RTU protocol encapsulated in a TCP/IP wrapper. MODBUS RTU is used for serial communications between a master and slave(s) devices. MODBUS TCP/IP is used for TCP/IP communications between client and server devices on an Ethernet network. The TCP/IP version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

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 type 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

Client / Server Model



Protocol Description

A typical MODBUS TCP/IP 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 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 Output Table	yes	yes
02	Read Input Table	yes	yes
03	Read Holding Registers (when addressing mode is 584/984, this function is used to access analog output registers)	yes	yes
04	Read Input Registers (when addressing mode is 584/984, this function is used to access analog input registers)	yes	no
05	Force Single Output	yes	no
06	Preset Single Registers	yes	yes
08	Loop back / Maintenance	yes	no
15	Force Multiple Outputs	yes	yes
16	Preset Multiple Registers	yes	yes

Network Server (slave) Operation

This section describes how other MODBUS TCP/IP clients on a network can communicate with an H0/H2/H4–ECOM100 that you have configured for MODBUS TCP/IP 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/IP server operation.

MODBUS Function Codes Supported

The H0/H2/H4–ECOM100 supports the following MODBUS function codes when acting as a MODBUS TCP/IP server.

MODBUS Function Code	Function	DL05/06/205 /405 Data Types Available
01	Read Output Table	Y, C, T, CT
02	Read 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 Output	Y, C, T, CT
06	Preset Single Registers	V
08	Loop back / Maintenance	
15	Force Multiple Outputs	Y, C, T, CT
16	Preset Multiple 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/IP 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** technical support website.

DL05 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)	256	X0 – X377	2048 – 2303	Input
Special Relays (SP)	512	SP0 – SP777	3072 – 3583	Input
Outputs (Y)	256	Y0 – Y377	2048 – 2303	Coil
Control Relays (C)	512	C0 – C777	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)	256	S0 – S377	5120 – 5375	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)	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 – 2560	Input
Special Relays (SP)	512	SP0 – SP777	3072 – 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
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)	128	V1000 – V1177	512 – 639	Input Register
V Memory, user data (V)	256 3072 4096	V400 – V677 V1400 – V7377 V10000 – V17777	256 – 511 768 – 3839 4096 – 8191	Holding Register

DL240 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 – 2367	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	3984 – 4063 4070 – 4095	Holding Register

DL250–1 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 4096	V1400 – V7377 V10000 – V17777	768 – 3839 4096 – 8191	Holding Register
V Memory, system (V)	256	V7400 – V7777	3840 – 4095	Holding Register

DL260 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 – X1777	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 – 767	Input Register
V Memory, user data (V)	256 3072 11264	V400– V777 V1400 – V7377 V10000 – V35777	256 – 511 768 – 3839 4096 – 15359	Holding Register
V Memory, system (V)	256	V7600 – V7777 V36000 – V37777	3968 – 4095 15360 – 16383	Holding Register

DL430 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)	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 – 511	Input
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)	3072	V1400 – V7377	768 – 3839	Holding Register
V Memory, system (V)	256	V7400 – V7777	3840 – 4095	Holding Register

DL440 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 – V1177	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)	288	V700 – V737 V7400 – V7777	448 – 479 3840 – 4095	Holding Register

DL450 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)	1024	X0 – X1777	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 (CR)	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 3840 – 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 (1089).
3. Use the MODBUS data type from the table.

PLC Address (Dec.) + Data Type

V2100 = 1088 decimal

1088 + Hold. Reg. = **Holding Reg. 1089**

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 (CR)	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 Address (Dec.) + Data Type

TA10 = 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 Type

C54 = 44 decimal

44 + 3073 + Coil = **Coil 3117**

Outputs (Y)	320	Y0	–	Y477	2048	–	2367	Coil
Control Relays (CR)	256	C0	–	C377	3073	–	3551	Coil

If the Host Software or Client Requires an Address ONLY

Some MODBUS TCP/IP 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** 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
		GX1747 – GX3777	11000–12048	
Inputs (X)	1024	X0 – X1777	12049 – 13072	
Special Relays (SP)	512	SP0– SP777	13073 – 13584	
Reserved	–	–	13585 – 20000	
Global Outputs (GY)	2048	GY0– GY3777	1 – 2048	Read/ Write
Outputs (Y)	1024	Y0 – Y1777	2049 – 3072	
Control Relays (CR)	2048	C0 – C3777	3073 – 5120	
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)	**QTY (Dec.)**	**PLC Range (Octal)**	**MODBUS 40001 Address Range**	**MODBUS 30001 Address Range**	**Access**
V Memory (Timers)	256	V0 – V377	40001 – 40256	30001 – 30256	Read/ Write
V Memory (Counters)	256	V1000 – V1377	40513 – 40768	30513 – 30768	
V Memory (Data Words)	256	V400 – V777	40257 – 40512	30257 – 30512	
	3072	V1400 – 7377	40769 – 43840	30769 – 33840	
	5903	V10000 – V23416	44097 – 49999	34097 – 39999	
	5361	V23417 – V35777	410000 – 415360	310000 – 315360	
V Memory (System Parameters)	128	V7600 – V7777	43969 – 44096	33969 – 34096	
	1024	V36000 – V37777	415361 – 416384	315361 – 316384	
V Memory (Remote Inputs)	128	V40000 – V40177	416385 – 416512	316385 – 316512	Read only
V Memory (Remote Outputs)	128	V40200 – V40377	416513 – 416640	316513 – 316640	Read/ Write
V Memory (Input Points)	64	V40400 – V40477	416641 – 416704	316641 – 316704	Read only
V Memory (Output Points)	64	V40500 – V40577	416705 – 416768	316705 – 316768	Read/ Write
V Memory (Control Relays)	128	V40600 – V40777	416769 – 416896	316769 – 316896	
V Memory (Timers Status Bits)	16	V41100 – V41117	416961 – 416976	316961 – 316976	
V Memory (Counter Status Bits)	16	V41140 – V41157	416993 – 417008	316993 – 317008	
V Memory (Special Relays)	32	V41200 – V41237	417025 – 417056	317025 – 317056	Read only

***** 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.

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 Address (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		3001	30001	Input Reg
Counter Current Values (V)	128	V1000 – V1177		512 – 639		3001	30001	Input Reg
V Memory, user data (V)	1024	V2000 – V3777		1024 – 2047		4001	40001	Hold 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

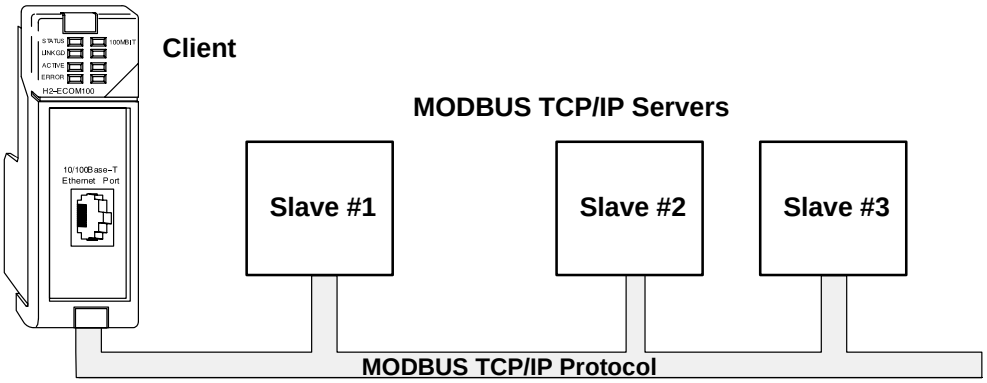
C54 = 44 decimal

44 + 3072 + 1 = **3117**

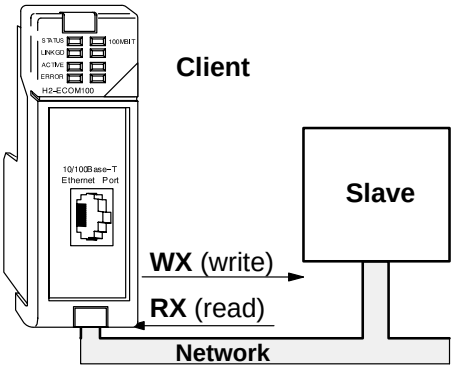
Outputs (Y)	320	Y0 – Y477	2048 – 2367	1	1	Coil
Control Relays (CR)	256	C0 – C377	3072 – 3551	1	1	Coil
Timer Contacts (T)	128	T0 – T177	6144 – 6271	1	1	Coil

Network Client (master) Operation

This section describes how the DL05/06/205/405 CPU can serve as a client on a MODBUS TCP/IP 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/IP client.

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

Note: The H0/H2/H4–ECOM100, as a client/master, does not support function code 4. Thus, 30001 address ranges cannot be read from a server/slave device.

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 – X, GX, SP
- Discrete – Y, CR, S, T, C
- 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 begining 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** 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
		GX1747 – GX3777	11000–12048	
Inputs (X)	1024	X0 – X1777	12049 – 13072	
Special Relays (SP)	512	SP0– SP777	13073 – 13584	
Reserved	–	–	13585 – 20000	
Global Outputs (GY)	2048	GY0– GY3777	1 – 2048	Read/ Write
Outputs (Y)	1024	Y0 – Y1777	2049 – 3072	
Control Relays (CR)	2048	C0 – C3777	3073 – 5120	
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	

Word Data Types*				
Registers (Word)	**QTY (Dec.)**	**PLC Range (Octal)**	**MODBUS Address Range**	**Access**
V Memory (Timers)	256	V0 – V377	40001 – 40256	Read/ Write
V Memory (Counters)	256	V1000 – V1377	40513 – 40768	
V Memory (Data Words)	256	V400 – V777	40257 – 40512	
	3072	V1400 – 7377	40769 – 43840	
	5903	V10000 – V23416	44097 – 49999	
	5361	V23417 – V35777	410000 – 415360	
V Memory (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 begining of the MODBUS word address range (40001) from the desired MODBUS address to target.
2. Convert decimal result into octal.
3. Add octal result to begining PLC range (Input, Output or Word).

1. 41025 – 40001 = 1024 decimal
2. 1024 decimal = 2000 octal
3. V0 (octal) + 2000 (octal) = **V2000** octal

Example 2: Calculating Discrete Input PLC Address

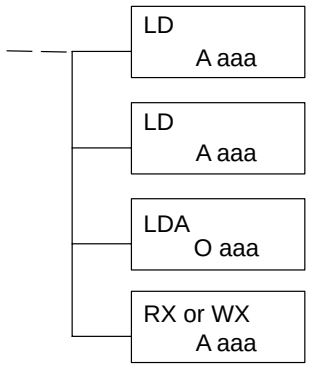
Find the PLC address to use to target MODBUS address **12060** in a server device.

1. Subtract the begining of the MODBUS Input address range (12049) from the desired MODBUS address to target.
2. Convert decimal result into octal.
3. Add octal result to begining PLC range (Input, Output or Word).

1. 12060 – 12049 = 11 decimal
2. 11 decimal = 13 octal
3 X0 (octal) + 13 (octal) = **X13** octal

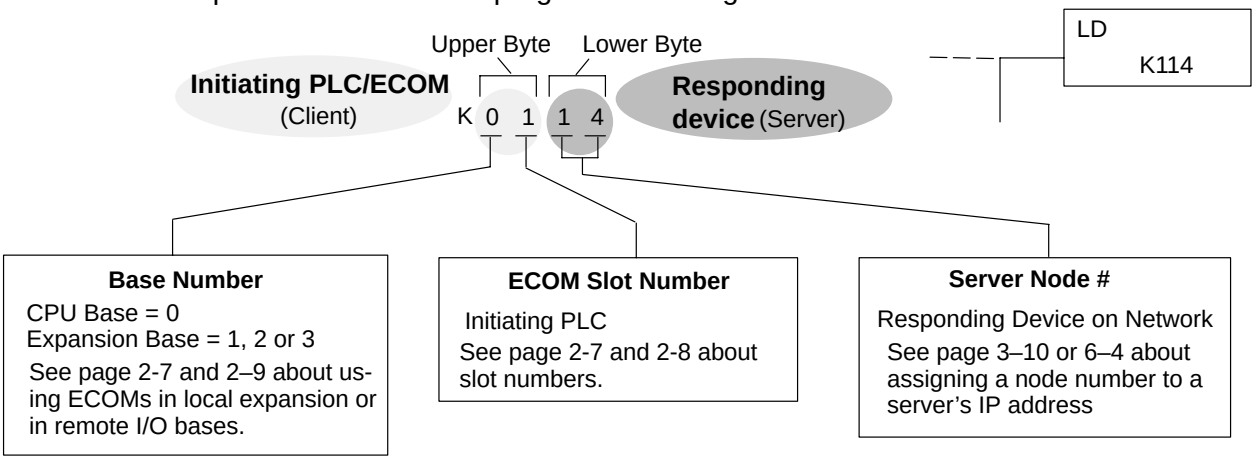
Building the Read (RX) or Write (WX) Routine

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.



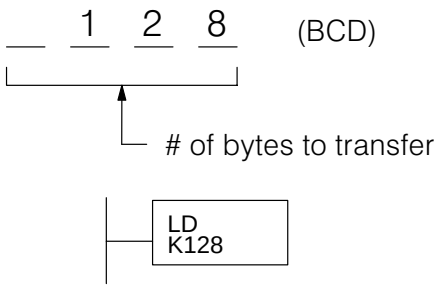
Step 1: Identify ECOM Slot Location and Server Node #

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.

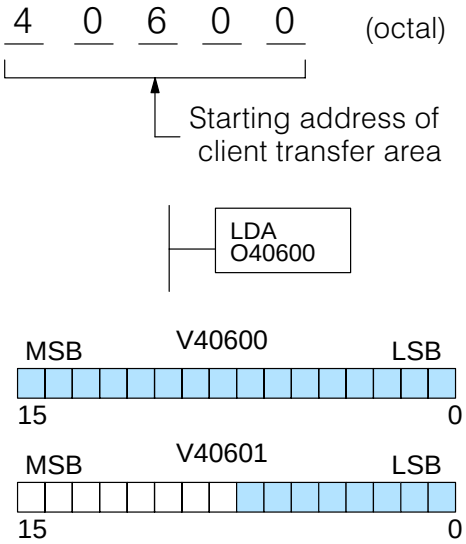


Step 3: Specify Master Memory Area

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.

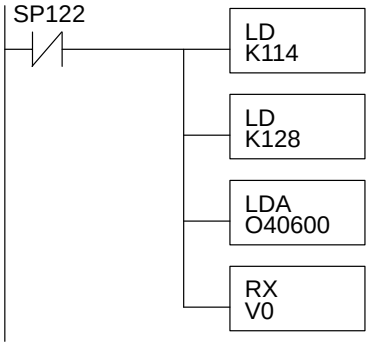
For an RX instruction, the CPU reads the number of bytes previously specified from the server, placing the received data into its memory area beginning at the LDA address specified.



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.

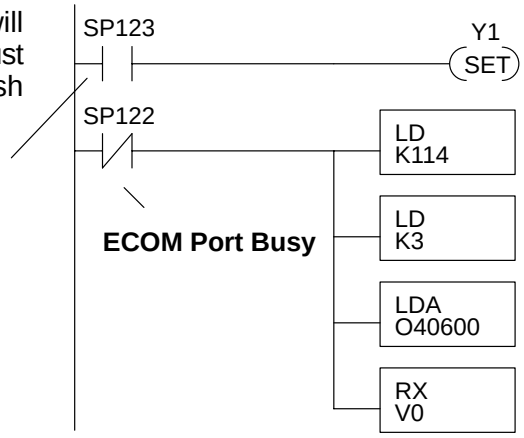
Step 4: Specify Slave Memory Area

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.



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 above 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.

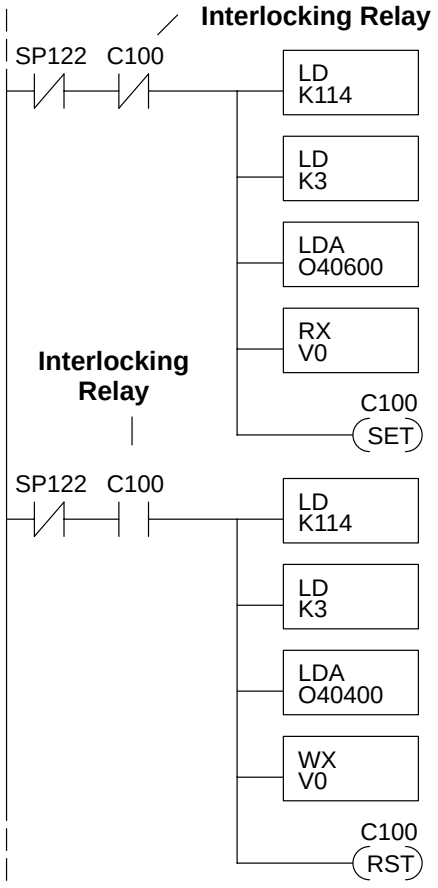
The “Port Communication Error” bit turns on when the PLC has detected an error. Use of this bit is optional. When used, it should be ahead of any network instruction boxes since the error bit is reset when an RX or WX instruction is executed.

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, C0 is set. When the port has finished the communication task, the second routine is executed and C0 is reset.

If you’re using RLL*PLUS* Stage Programing, 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.



H0/H2/H4–ECOM100 System Memory

	H0/H2/H4–ECOM100			
	Modbus Addressing 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.**