ETHERNET

Type:

System Command

Syntax:

```
ETHERNET(rw, slot, function[, parameters...])
```

Description:

The command ETHERNET is used to configure the operation of the Ethernet port.

> *Many of the ETHERNET functions are command line only; these are stored in flash EPROM and are then used on power up.*

Parameters:

rw:	Specifies the required action.	
	0	Read
	1	Write
slot:	Set to -1 for the built in Ethernet port	
function:	Function to be performed	
	0	IP Address
	1	Reserved function
	2	Subnet Mask
	3	MAC address (read only)
	4	Default Port Number
	5	Token Port Number
	6	PRP firmware version (MC464 read only)
	7	Modbus TCP mode
	8	Default Gateway

	9	Data configuration
	10	Modbus TCP port number
	11	ARP cache
	12	Reserved function
	13	Reserved function
	14	Configure endpoints for Modbus TCP or Ethernet IP
	15	Set the maximum packets handled per stack call
	21	List status of ports to the terminal
	22	Read Ethernet port MIB information
	23	Reserved for device specific PHY registers
	24	Read Ethernet connection state

Function = 0: IP Address

Prints or writes the Ethernet IP address. This is command line only.

You must power cycle the controller or perform EX(1) to apply the new IP address.

```
ETHERNET(rw, slot, 0[, byte1, byte2, byte3, byte4])
```

Alternative:

```
  IP_ADDRESS = xxx.xxx.xxx.xxx
```

Parameters:

byte1:	The first byte of the IP address
byte2:	The second byte of the IP address
byte3:	The third byte of the IP address
byte4:	The fourth byte of the IP address

The default address is 192.168.0.250

Example:

Read the current IP address and then set a new IP address into the controller and perform an EX(1) to activate the address

Performing an EX(1) as in this example will close the communications and you will only be able to communicate again using the new IP address.

```
>>ETHERNET(0, -1, 0)
192.168.0.250
>>ETHERNET(1, -1, 0, 192, 168, 0, 201)
>>EX(1)
>>
```

Function = 2: Subnet Mask

Prints or writes the Subnet Mask. This is command line only.

You must power cycle the controller or perform EX(1) to apply the new IP address.

```
ETHERNET(rw, slot, 2[, byte1, byte2, byte3, byte4])
```

Alternative:

```
IP_NETMASK = xxx.xxx.xxx.xxx
```

Parameters:

byte1:	The first byte of the Subnet Mask
byte2:	The second byte of the Subnet Mask
byte3:	The third byte of the Subnet Mask
byte4:	The fourth byte of the Subnet Mask

The default Subnet Mask is 255.255.255.0

Example:

Read the subnet mask and write a new value

```
>>ETHERNET(0, -1, 0)
255.255.255.0
>>ETHERNET(1, -1, 2, 255, 255, 128, 0)
>>
```

Function = 3: MAC Address

Prints the MAC address. This is command line only.

This function is read only.

```
ETHERNET(0, slot, 3)
```

Parameters:

The MAC address is unique to your controller.

Example:

Read the MAC address of a controller

```
>>ETHERNET(0, -1, 3)
00:06:70:00:00:FA
>>
```

Function = 4: Default Port

Prints or writes the default port number. This is command line only.

> *The default value is used by Motion Perfect and PCMotion and should not be changed unless absolutely necessary.*

`ETHERNET(rw, slot, 4[, port])`

Parameters:

port:	The port used for the main command line in the controller. (default 23)

Function = 5: Token Port

Prints or writes the default port number for token channel which is used by the PCMotion ActiveX control. This is command line only.

> *The default value is used by the PCMotion ActiveX control and should not be changed unless absolutely necessary.*

`ETHERNET(rw, slot, 5[, port])`

Parameters:

port:	The port used for the token channel in the controller. (default 3240)

Function = 6: PRP Firmware Version (read only)

Reads the communications processor’s firmware version. This is command line only.

> *This function is read only*

`ETHERNET(0, slot, 6)`

Parameters:

Returns the flash application version and the bootloader version.

Example:

Read the communications processor firmware with application version 61 and boot loader version 22.

```
>>ETHERNET(0, -1, 6)
```

```
61;22
>>
```

Function = 7: Modbus TCP mode

Sets the Modbus TCP data type. This value is stored in RAM and so must be initialised every time the controller powers up. This can be done in a TrioBASIC program for example STARTUP

This must be configured before the Modbus master opens the port.

```
ETHERNET(rw, slot, 7[, mode])
```

Parameters:

mode:	0	16-bit integer (default value)
	1	32-bit single precision floating point without address halving
	2	32-bit long word integers without address halving

If you want to use address halving please see ETHERNET Function 14

Example:

Initialise the Modbus TCP port for floating point data.

```
ETHERNET(1, -1, 7, 1)
```

Function = 8: Default Gateway

Prints or writes the Default Gateway. This is command line only.

You must power cycle the controller or perform EX(1) to apply the new Default Gateway.

```
ETHERNET(rw, slot, 8[, byte1, byte2, byte3, byte4])
```

Alternative:

```
IP_GATEWAY = xxx.xxx.xxx.xxx
```

Parameters:

byte1:	The first byte of the Default Gateway
byte2:	The second byte of the Default Gateway
byte3:	The third byte of the Default Gateway
byte4:	The fourth byte of the Default Gateway

Example:

Print then change the value of the default gateway.

```
>>ETHERNET(0, -1, 8)
192.168.0.225
>>ETHERNET(0, -1, 8, 192, 168, 0, 150)
>>
```

Function = 9: Data configuration

Sets the Modbus TCP data source. This value is stored in RAM and so must be initialised every time the controller powers up. This can be done in a TrioBASIC program for example STARTUP

This must be configured before the Modbus master opens the port.

```
ETHERNET(rw, slot, 9[, mode])
```

Parameters:

mode:	0	VR (default value)
	1	Table

Example:

Initialise the Modbus TCP port for table data.

```
ETHERNET(2, -1, 9, 1)
```

Function = 10: Modbus TCP port number

Prints or writes the default port number for token channel which is used by Modbus TCP. This is command line only.

The default value is used by Modbus and should not be changed unless absolutely necessary.

```
ETHERNET(rw, slot, 10[, port])
```

Parameters:

port:	The port used for the token channel in the controller. (default 502)

Function = 11: ARP cache

Reads the ARP cache. (Command line only.)

This function is read only

```
ETHERNET(0, slot, 11)
```

Function = 14: Endpoints for Modbus TCP or Ethernet IP

This function allows the user to configure Ethernet IP and Modbus at a low level. The default values allow a master to connect without any configuration on the Controller side. These settings are stored in RAM and so must be initialised every time the controller powers up. This can be done in a TrioBASIC program for example STARTUP.

As mentioned above, both, Ethernet IP and Modbus TCP would work without any configuration (using default values) on power-up. In case a different initial configuration is needed, the system parameter IP_PROTOCOL_CTRL must be configured in the MC_CONFIG file in order to disable the protocol and set the configuration required within a BASIC start-up routine.

```
ETHERNET(1, slot, 14, endpoint_id, parameter_index,
parameter_value )
```

Parameters:

endpoint_id:	This allows you to specify which end point you are reading or writing	
	0	Modbus TCP Server (Slave)
	1	Ethernet IP Assembly Object, Instance 100 (input)
	2	Ethernet IP Assembly Object, Instance 101 (output)
	3	Ethernet IP Assembly Object, Instance 102 (input)
	4	Ethernet IP Assembly Object, Instance 103 (output)
	6	Modbus TCP Client (Master)
parameter_index:	This parameter selects which of the endpoint variables you are reading or writing	
	0	Address
	1	Data location
	2	Data format
	3	Length
	4	Class
	5	Instance

	6	Operation Mode
	7	Set/read memory map
parameter_value:	Dependent on Parameter index, see table below	

Parameter values:

parameter_index	**parameter_value**	
0	The start position of the data location.	
1	The location of the data on the controller.	
	0	Register (reserved use)
	1	I/O input
	2	I/O output
	3	VR (default value)
	4	Table
	5	Digital I/O Input
	6	Digital I/O Output
	7	Analogue I/O Input
	8	Analogue I/O Output
2	The precision of the data.	
	0	Integer 16 bit (default value)
	1	Integer 32 bit
	2	Floating point 32 bit
	3	Floating point 64 bit
	4	Unsigned integer 16 bit
3	The number of the data locations returned.	

4	The class. This function is read only.		
	4	Ethernet IP	
	68	Modbus	
5	The instance of the endpoint. This function is read only.		
	0	Modbus	
	100	Ethernet IP input	
	101	Ethernet IP output	
6	The Operation mode. Read/write.		
	0	Modbus TCP uses normal addressing	
	1	Modbus TCP uses “address halving”	
7	Set memory mapping for Modbus TCP		
	Request address	Modbus request address. If request address of -1 is used, then all defined regions are reset	
	Base address	Base address on the controller memory area to which the request address is mapped	
	Num entries	Number of entries in the mapped region (mapped address area is from the base address to (base address + num_entries – 1). Minimum value is 1	
	Mem area	0	Map Modbus data to VR
		1	Map Modbus data to TABLE
	Data type	0	Integer 16
		1	Integer 32
		2	Floating Point 32

Note: address halving will be automatically used for 32 bit data types when mapped with parameter_index 7.

Examples:

Example 1:

Configure Modbus using Function 14 to use Table and floating point 64-bit

```
ETHERNET(1, -1, 14, 0, 1, 4)
ETHERNET(1, -1, 14, 0, 2, 3)
```

Example 2:

Configure Ethernet IP for 50 TABLE inputs starting at 200 and 50 table outputs starting at 300 all at 32-bit float

```
'Inputs
ETHERNET(1, -1, 14, 1, 0, 200)
ETHERNET(1, -1, 14, 1, 1, 4)
ETHERNET(1, -1, 14, 1, 2, 2)
ETHERNET(1, -1, 14, 1, 3, 50)
'Outputs
ETHERNET(1, -1, 14, 2, 0, 300)
ETHERNET(1, -1, 14, 2, 1, 4)
ETHERNET(1, -1, 14, 2, 2, 2)
ETHERNET(1, -1, 14, 2, 3, 50)
```

Example 3:

Configure Modbus TCP floating point TABLE access, using address halving to match the addressing scheme used in the master.

```
ETHERNET(1, -1, 14, 0, 2, 2)
ETHERNET(1, -1, 14, 0, 1, 4)
ETHERNET(1, -1, 14, 0, 6, 1)
```

Example 4:

Configure Modbus TCP mapping to VR memory with zones representing different data types. Modbus registers 100-109, 110-128, 130-148 and 150-168 are mapped.

32 bit values require 2 x 16 bit registers per value.

```
ETHERNET(1, -1, 14, 0, 7, -1) 'Cancel all previous maps
ETHERNET(1, -1, 14, 0, 7, 100, 200, 10, 0, 0) 'VRs 200 to 209
size = Int 16
ETHERNET(1, -1, 14, 0, 7, 110, 300, 10, 0, 1) 'VRs 300 to 209
size = Int 32
ETHERNET(1, -1, 14, 0, 7, 130, 400, 10, 0, 2) 'VRs 400 to 409
size = FP 32
ETHERNET(1, -1, 14, 0, 7, 150, 400, 10, 1, 2) 'TABLE 400 to 409
size = FP 32
ETHERNET(0, -1, 14, 0, 7) 'Display the map to terminal #0
```

Map displayed in the terminal:

```
0: Req: 100, Base: 200, Num: 10, Area: VR(3), DataType:
INT16(0)
1: Req: 110, Base: 300, Num: 10, Area: VR(3), DataType:
INT32(1)
2: Req: 130, Base: 400, Num: 10, Area: VR(3), DataType: FP32(2)
3: Req: 150, Base: 400, Num: 10, Area: Table(4), DataType:
FP32(2)
4: Req: 0, Base: 0, Num: 0, Area: Unknown(0), DataType:
INT16(0)
5: Req: 0, Base: 0, Num: 0, Area: Unknown(0), DataType:
INT16(0)
6: Req: 0, Base: 0, Num: 0, Area: Unknown(0), DataType:
INT16(0)
7: Req: 0, Base: 0, Num: 0, Area: Unknown(0), DataType:
INT16(0)
```

Function = 15: Maximum packets handled per stack call

ARM based controllers only.

Function '15' of the Ethernet command provides access to a parameter which limits the number of incoming Ethernet packets handled per call to a particular stack function. It controls the rate, and the smaller the value the slower the rate of packet handling but the less chance of overloading other processes going on in the microprocessor.

The default value is 0, which means handle all incoming packets. This parameter is dynamic and must be set each time from power on in a program.

```
ETHERNET(r/w, slot, 15, value)
```

Example 1:

In the terminal 0 command line, check the setting of function 15.

```
>>?ETHERNET(0, -1, 15)
0
-1
```

Example 2:

In a program, set the max number of packets that can be handled in one stack call to 20. This will be enough to limit the impact of large data transfers over the TrioPC ActiveX from affecting motion and other networks.

```
ETHERNET(1, -1, 15, 20)
```

Function = 21: List port status

Prints a report to the current output stream showing connection details for the Ethernet ports.

```
ETHERNET(0, slot, 21)
```

Example:

```
>>ETHERNET(0, -1, 21)
Socket: index 13 is connected.
Socket: index 13, Local IP: 192.168.0.249 Port: 502
Socket: index 13, Remote IP: 192.168.0.57 Port: 56683
Socket: keep alive cfg: Interval: 20, Idle: 20, Count: 8,
Socket: keep alive : value: 20, Abort: -1

Socket: index 14 is connected.
Socket: index 14, Local IP: 192.168.0.249 Port: 3240
Socket: index 14, Remote IP: 192.168.0.142 Port: 1297
Socket: keep alive cfg: Interval: 20, Idle: 20, Count: 8,
Socket: keep alive : value: 20, Abort: -1

Socket: index 15 is connected.
Socket: index 15, Local IP: 192.168.0.249 Port: 23
Socket: index 15, Remote IP: 192.168.0.142 Port: 1271
Socket: keep alive cfg: Interval: 20, Idle: 20, Count: 8,
Socket: keep alive : value: 20, Abort: -1
```

Port 502 is the Modbus TCP port, 3240 is for TrioPC Motion ActiveX/DLL and 23 is for Motion Perfect.

Function = 22: Read MIB information

This command should be used in the command line terminal only. Function 22 returns the MIB information as text to the command line output.

Example:

In the terminal, type the command and read the text response.

```
>>ETHERNET(0,-1,22)
MIB Information:
RMON_T_DROP : 0
RMON_T_PACKETS : 18322
RMON_T_BC_PKT : 259
Etc.
```

Function = 24: Read connection state

Returns the cable connection state as TRUE/FALSE. If the RJ45 connector has a live Ethernet connection plugged in then function 24 returns TRUE. If the cable is unplugged, or is connected to a switch or other device that is powered off, then the function returns FALSE.

```
DIM state AS BOOLEAN
state = ETHERNET(0, slot, 24)
```

Example:

```
' Monitor the Ethernet connection and flash an output when
' connection is lost

WHILE TRUE
  IF ETHERNET(0, -1, 24) = FALSE THEN
    OP(9, 0)
    OP(8, 1) ' Output 8 flashing shows connection lost
    WA(250)
    OP(8, 0)
    WA(250)
  ELSE
    (9, 1) ' Output 9 shows a good connection
    WA(250)
  ENDIF
WEND
```

See also:

MODBUS, IP_PROTOCOL_CTRL, IP_TCP_TX_THRESHOLD, IP_TCP_TX_TIMEOUT