ETHERCAT

Type:

System Command

Syntax:

```
ETHERCAT(function, slot [,parameters…])
```

Description:

The command ETHERCAT is used to perform advanced operations on the EtherCAT network. In normal use the EtherCAT network will start automatically without the need for any commands in a startup program. Some ETHERCAT command functions may be useful when debugging and setting up an EtherCAT system, so a small sub-set is described here.

The ETHERCAT command returns TRUE(-1) if successful and FALSE (0) if the command execution was in error. Functions which return a value must either put the value in a VR or print it to the current output terminal.

Parameters:

function:	Function to be performed	
	$00	Start EtherCAT network
	$01	Stop EtherCAT network
	$02	Check slave online
	$03	Check number of slaves
	$04	Get slave address
	$05	Get slave axis
	$21	Set EtherCAT State
	$22	Get EtherCAT State
	$30	Set Slave ESC Register value (Auto)
	$31	Get Slave ESC Register value (Auto)

	$64	Send fault reset sequence to a drive
	$79	Set Status Mask and Trigger Pattern
	$7A	Get Status Mask and Trigger Pattern
	$87	Display network configuration
	$91	Set Sync0 Start time
	$93	Set DC Master mode
	$E0	Set Emergency Message control
	$E1	Read Emergency Message control value
slot:	Set to the EtherCAT port slot number	
	MC4N, MC6N	0
	Flex-6 Nano	0
	MC464	P876 slot number (0 to 7)
	MC664	Internal slot number or -2 P876 slot number if fitted

Function = $00: Start EtherCAT network

Syntax:

```
ETHERCAT(0, slot, [,option_flags [,MAC_retries]])
```

Description:

Initialise EtherCAT network, and put it onto operational mode.

Parameters:

slot:	Slot number of EtherCAT port.		
option_flags:	bit	state	Description
	0	0	Do not print progress information
		1	Print out progress information (default)

	1	0	Normal state change (default)
		1	Force a re-scan of the network
	2	0	Standard startup
		1	Express startup; do not wait for PLL to lock between state changes
MAC_retries:	Sets the number of times the master attempts to restart the Ethernet auto-negotiation. Default = 2		

Example:

Check for the EtherCAT state and if not in Operational State, restart the EtherCAT and set an output to indicate that a re-start is in progress.

```
'--Init EtherCAT if needed.
Slt = 0
ecs_vr = 30 'Use VR 30 for returned value
chk = ETHERCAT($06, slt, ecs_vr) 'Test state

IF chk <> TRUE OR VR(ecs_vr) <> 3 THEN
  OP(9, ON)
  WA(15000) ' wait 15sec for drive to power up
  ETHERCAT(0, slt) 'Initialise EtherCAT
ENDIF
```

Function = $01: Stop EtherCAT network

Syntax:

```
ETHERCAT(1, slot)
```

Description:

Closedown the EtherCAT network.

Parameters:

None.

Example:

Stop the EtherCAT protocol from the terminal and then re-start it.

```
>>ETHERCAT(1, 0)
>>ETHERCAT(1, 0)
>>
```

Function = $02: Check slave online

Syntax:

```
ETHERCAT(2, slot, slave_position, vr_index)
```

Description:

Check whether given slave is online. If the vr_index <> -1, then write the result into the given VR. If the vr_index = -1 then write the result to the standard output stream. Return whether the function was successful.

Slave must have been initialised and on the network before the status can be checked.

Parameters:

slave_position:	Position on the network of the slave to be checked.	
vr_index:	0 - VRmax	Result is placed in the selected VR variable.
	-1	Result is displayed in the current output path. (default terminal 0)

Example:

Check if the slaves at network positions 0 to 7 have gone offline.

```
FOR posn = 0 TO 7
  ETHERCAT(2, 0, posn, 20 + posn)
  IF VR(20 + posn) = FALSE THEN
    PRINT "Drive at "; posn[0]; " is offline"
  ENDIF
NEXT posn
```

Function = $03: Check number of slaves

Syntax:

```
ETHERCAT(3, slot, vr_index)
```

Description:

Determine the number of slaves on the bus. If the vr_index <> -1, then write the result into the given VR. If the vr_index = -1 then write the result to the standard output stream. Return whether the function was successful.

Parameters:

vr_index:	0 - VRmax	Result is placed in the selected VR variable.
	-1	Result is displayed in the current output path. (default terminal 0)

Example:

Check to see how many slaves are on the EtherCAT bus.

```
    ETHERCAT(3, 0, 20)
    PRINT "There are "; VR(20)[0]; " slaves on the EtherCAT bus"
```

Function = $04: Get slave address

Syntax:

```
ETHERCAT(4, slot, slave_position, vr_index)
```

Description:

Determine the configured address of the slave identified by its position on the network. If vr_index <> -1, then write the result into the given VR. If vr_index = -1 then write the result to the standard output stream. Return whether the function was successful.

Slave must have been initialised and on the network before the configured address can be checked.

Parameters:

slave_position:	Position on the network of the slave to be checked.	
vr_index:	0 - VRmax	Configured address is placed in the selected VR variable.
	-1	Configured address is displayed in the current output path. (default terminal 0)

Example:

Report the configured addresses of every node on the network.

```
    ETHERCAT(3, 0, 20) 'Get the number of slaves on the bus
    FOR posn = 0 TO VR(20) - 1
      ETHERCAT(4, 0, posn, 30 + posn)
      PRINT "Drive at position "; posn[0]; " has address ";
  VR(30+posn)[0]
    NEXT posn
```

Function = $05: Get slave axis

Syntax:

`ETHERCAT(5, slot, slave_position, vr_index)`

Description:

Determine the axis of the slave identified by its position on the network. If vr_index <> -1, then write the result into the given VR. If vr_index = -1 then write the result to the standard output stream. Return whether the function was successful.

> *Slave must have been initialised and on the network before the Axis number can be checked.*

Parameters:

slave_position:	Position on the network of the slave to be checked.	
vr_index:	0 - VRmax	Axis number is placed in the selected VR variable.
	-1	Axis number is displayed in the current output path. (default terminal 0)

Example:

Report the configured addresses of every node on the network.

```
    ETHERCAT(3, 0, 20) ' get the number of slaves on the bus
    FOR posn = 0 TO VR(20) - 1
      ETHERCAT(5, 0, posn, 30 + posn)
      PRINT "Drive at position "; posn[0]; " has Axis ";
  VR(30+posn)[0]
    NEXT posn
```

Function = $21: Set EtherCAT State

Syntax:

`ETHERCAT($21, slot, state, display)`

Description:

This function controls the EtherCAT State Machine. (ESM) It requests the master change to given EtherCAT ‘state’, and hence changes all slaves to the same state. When a change to a higher state is made, the EtherCAT network will progress to the new state through the in-between states to allow correct starting of the network.

Parameters:

state:	EtherCAT state request	
	-1	Reserved
	0	Initial (EtherCAT ESC value 0x01)
	1	Pre-Operational (0x02)
	2	Safe-Operational (0x04)
	3	Operational (0x08)
display:	Function	
	1	Writes state change information to the standard output stream. (Default)
	0	Do not write out state change information.

Example:

Change the EtherCAT to Safe-Operational and suppress the information that would be printed to the terminal.

```
  ETHERCAT($21, 0, 2, 0)
```

Function = $22: Get EtherCAT State

Syntax:

```
ETHERCAT($22, slot, vr_number)
```

Description:

Gets the present state of the EtherCAT running on the defined slot. The value returned shows the EtherCAT state as follows:

- 0 – Initial
- 1 – Pre-oprational
- 2 – Safe-Operational
- 3 - Operational

Parameters:

vr_number:	The VR number where the returned value will be put.

	(-1 forces the value to be printed on the terminal)

Example:

In the terminal, request the EtherCAT state value.

```
>>ETHERCAT($22, 0, -1)
3
>>
```

Function = $30: Set Slave ESC Register (Auto)

Syntax:

```
ETHERCAT($30, slot, slave_address, ESC_offset, data_length,
value)
```

Description:

Low-level write to the slave's ESC registers, addressing method used is auto-increment addressing.

Returns TRUE if the write is successful.

Parameters:

slot:	Slot number in the Motion Coordinator
slave_address:	Auto-increment address
ESC_offset:	Register address
data_length	Number of bytes in the register size
value, value1	Value/value1 are 32-bit data values, LS word first.

Example:

In the terminal, set the value of ESC register $0935 (speed counter filter depth) in slave at address number 2.

```
>>?ETHERCAT($30, 0, 2, $0935, 2, 12)
-1
```

Function = $31: Get Slave ESC Register (Auto)

Syntax:

```
ETHERCAT($31, slot, slave_address, ESC_offset, data_length,
vr_index)
```

Description:

Low-level read from the slave's ESC registers, addressing method used is auto-increment addressing.

Returns TRUE if the read is successful.

Parameters:

slot:	Slot number in the Motion Coordinator		
slave_address:	Auto-increment address		
ESC_offset:	Register address		
data_length	Number of bytes in the register size		
vr_index	-1	Display the value on the current default output channel	e.g. terminal #0
	0 .. max_vr_index-1	Store the value in the VR number specified.	

Example:

In the terminal, read the value of ESC register $0130 (EtherCAT Slave State) in slave at address number 5. Print the result to the terminal.

```
>>?ETHERCAT($31, 0, 5, $0130, 2, -1)
8
-1
```

Function = $64: Send reset sequence to a drive

Syntax:

```
ETHERCAT($64, axis_number[, mode[, timeout]])
```

Description:

Reset a slave error. This function runs the error reset sequence on the drive control word. DRIVE_CONTROLWORD bit 8 is toggled high then low. This will instruct the drive to reset any errors in the drive where the cause of the error has been removed.

The response to a reset sequence will depend on the drive and how closely it follows the CoE CiA402 specification.

Parameters:

axis_number:	The axis number of the drive to be reset.

mode:	0	The 'Fault Reset' (bit 7) of DS402 control word is set high and then set low again after a hard coded timeout. (default)
	1	Bit 7 is set high until the 'Fault Flag' (bit 3) of the status word goes low, or a timeout occurs.
timeout:	Optional timeout in ms used during mode 1 operation. Default is 100 ms. Range is 1 to 10000 ms.	

Example:

Example 1

Send control word reset sequence to drive at axis 8.

```
ETHERCAT($64, 8)
```

Example 2

Send control word reset sequence to drive at axis 2. Use Mode 1 to force the reset bit to remain high until the status it 3 goes low or force the reset bit low again after 60 ms, even if the status bit is still high.

```
ETHERCAT($64, 2, 1, 60)
```

See Also:

UNIT_CLEAR, DATUM(0), AXIS_STATUS, DRIVE_STATUS

ETHERCAT $79 : Set Status Mask and Trigger Pattern

Type:

EtherCAT configuration setup. Runtime definition of slave – controller interaction.

Syntax:

```
ETHERCAT($79, slot, addr, vr_index [, mask, trigger_pattern])
```

Description:

This command enables the user to define a CoE Statusword (0x6041:00) mask and trigger pattern for raising the controller AXISSTATUS 'Remote Drive Status word Error' flag (bit 26.)

The mask defines which bits within the CoE status word are relevant, and the trigger pattern defines the state of these bits (0 or 1) which will trigger the controller AXISSTATUS flag.

If it is necessary to reset the watchdog (master enable) when this AXISISTATUS flag is set then the axis ERRORMASK must be set accordingly.

The slave mask and pattern can be set and modified from preop to operational state of the network (after the configured addresses have been assigned.) It must be reset if the network is restarted.

The slave must support the CoE statusword (0x6041:00) within the PDO cyclic data.

If the Mask is zero the controller will not execute any runtime check of the CoE statusword.

If the watchdog must be cleared to OFF when the 'remote drive statusword error' flag is raised then bit 26 of the axis ERRORMASK parameter must be set.

Parameters:

Slot:	module slot number		e.g. 0
Addr:	configured address of the slave		
vr_index:	-1	use the given mask and trigger_pattern values in parameters 5 and 6.	
	0 .. max_vr_index-1	The index of the VR which contains the mask and the next VR (vr_index+1) contains the trigger_pattern.	
Mask:	CoE statusword mask		Optional
Trigger_pattern:	Defines the state of the relevant bits which will trigger the controller AXISSTATUS 'remote drive statusword error' flag (bit 26.)		Optional

Return Value

-1 : mask and pattern set correctly

0: failed to set the mask and pattern

Example:

We want to set the AXISSTATUS 'remote drive statusword error' (bit 26) flag and drop the watchdog to OFF when the slave drive either sets the 'switch on disabled' flag (bit 6 of the CoE statusword), or clears the 'Ready to switch on' flag (bit 0 of the CoE statusword.)

Hence

```
'Mask = 0x41
'Trigger Pattern = 0x40

'Set mask and trigger pattern.
'Slot = 0, slave configured address = 1 and axis = 0

'Params(func, slot, addr, vr_index, mask, trigger pattern)
ETHERCAT($79, 0, 1, -1, $41, $40)

'Set ERRORMASK to ensure the AXISSTATUS bit 26 will clear the
WDOG when triggered.
BASE(0)
ERRORMASK.26 = 1
```

Now if the CoE status word bit 0 (ready to enable) is set low, or bit 6 (switch on disabled) is set high the AXISSTATUS ‘remote drive status word error’ flag (bit 26) shall be set and the WDOG cleared.

ETHERCAT $7A : Get Status Mask and Trigger Pattern

Type:

EtherCAT configuration setup. Runtime definition of slave – controller interaction.

Syntax:

```
ETHERCAT($7A, slot, addr[, vr_index])
```

Description:

This command enables the user to display and check the mask and trigger pattern values set (if any) for a given slave.

Parameters:

Slot:	module slot number		e.g. 0
Addr:	configured address of the slave		
vr_index:	-1	Display the mask and trigger_pattern values to the current default output channel.	e.g. terminal 0
	0 .. max_vr_index-1	Index of the array of 2 VRs where the mask and the trigger_pattern will be stored.	

Return Value

-1 : mask and pattern read correctly

0: failed to get the mask and pattern

Example:

```
' Mask and pattern have not been set, so default values
returned.
' Params(func, slot, addr [, vr_index])
>>?ETHERCAT($7A, 0, 1, -1)
0
0
-1

' Set Mask to0x41 and trigger pattern to 0x40, for slave on
slot 0 with configured addr of 1
>>ETHERCAT($79, 0, 1, -1, $41, $40)

' Read back the values (in decimal)
>>?ETHERCAT($7A, 0, 1, -1)
65
64
-1
```

Function = $87: Display network configuration

Syntax:

```
ETHERCAT($87, slot)
```

Description:

Displays the network configuration to the command line terminal in Motion Perfect.

Parameters:

slot:	The slot number where the EtherCAT module is located

Example:

In the terminal, request the EtherCAT network configuration.

```
>>ethercat($87,0)
EtherCAT Configuration (0):
    EK1100       :  0 : 0 : 2000
    EL2008       :  1 : 0 : 1000 (0:0/16:8)
    EL2008       :  2 : 0 : 1001 (0:0/24:8)
    EL2008       :  3 : 0 : 1002 (0:0/32:8)
    EL2008       :  4 : 0 : 1003 (0:0/40:8)
    EL2008       :  5 : 0 : 1004 (0:0/48:8)
    EK1110       :  6 : 0 : 2001
    RS2          :  7 : 0 : 1 (0)
    SGDV          :  8 : 0 : 2 (1)
>>
```

ETHERCAT $91 : Set EtherCAT Sync0 start time

Type:

EtherCAT configuration setup.

Syntax:

```
ETHERCAT($91, slot, vr_index[, value])
```

Description:

Set the network default Sync0 start time offset. If the vr_index <> -1, then use the value from the given VR, if the vr_index = -1 then write the value in the third parameter.

The default start time is 1 second (1000000 microseconds). Add or subtract the required offset in microseconds. For example 1000150 is an offset of 150 µs and 999930 is an offset of -70 µs. The offset must not exceed ±50% of the servo period.

The function returns TRUE/FALSE depending upon whether the operation was successful.

This command must be run before the EtherCAT network is started.

Parameters:

Slot:	module slot number	
vr_index:	-1	Use the value from parameter 3.
	0 .. max_vr_index-1	Use the value in the VR number specified.
value	Optional direct value for the start time.	

Example:

```
'Set the EtherCAT Sync0 start time with offset of 500
microseconds.
e_flag = ETHERCAT($91, 0, -1, 1000500)
IF e_flag = FALSE THEN
    PRINT #5,"Failed to set Sync0 start time"
ENDIF
```

ETHERCAT $93 : Set Distributed Clocks master mode

Type:

EtherCAT configuration setup.

Syntax:

```
ETHERCAT($93, slot, dc_function, vr_index[, value])
```

Description:

Set the EtherCAT master to enable the first DC enabled node to be the Distributed Clocks master. As this requires the Trio Motion Coordinator to synchronise to the external DC master, it does not function on P876, MC4N-ECAT or MC664(X) as these have FPGA based fixed timing hardware for the EtherCAT port.

The function returns TRUE/FALSE depending upon whether the operation was successful.

This command must be run before the EtherCAT network is started.

Parameters:

Slot:	module slot number	
dc_function:	Distributed Clocks function. 2 = use first DC enabled node as clocks master.	
vr_index:	-1	Use the value from parameter 3.
	0 .. max_vr_index-1	Use the value in the VR number specified.
value	Optional direct value for the DC mode. 0 = use first DC enabled node as the clocks master.	

Example:

Set the first drive with DC capability to be the Distributed Clocks Master then start the network.

```
'Use first DC enabled drive as master clock
ETHERCAT($93, 0, 2, -1, 0)
'Start EtherCAT network
ETHERCAT(0, 0)
```

Function = $E0: Set Emergency Message control

Syntax:

```
ETHERCAT($E0, slot, mode)
```

Description:

Set the response of the master to an emergency telegram transmitted by a slave node. This function enables the user to determine whether emergency messages should trigger a system error alarm and cause the WDOG to be dropped.

Parameters:

mode:	0	Log error and raise a System Error and drop WDOG off. (Default)
	1	Log error only.

Example:

Set the mode so that Emergency Messages from slave nodes do not cause the WDOG to go OFF. Then start the EtherCAT network.

```
ec_slot = 0
ETHERCAT($E0, ec_slot, 1)
ETHERCAT(0, ec_slot)
```

Function = $E1: Read Emergency Message control value

Syntax:

```
ETHERCAT($E1, slot, vr_number)
```

Description:

Get the current value; indicating the response of the master to an emergency telegram transmitted by a slave node.

Parameters:

vr_number:	The VR number where the returned value will be put. (-1 forces the value to be printed on the terminal)

Example:

In the terminal, request the EtherCAT Emergency Message response setting.

```
>>ETHERCAT($E1, 0, -1)
1
>>
```