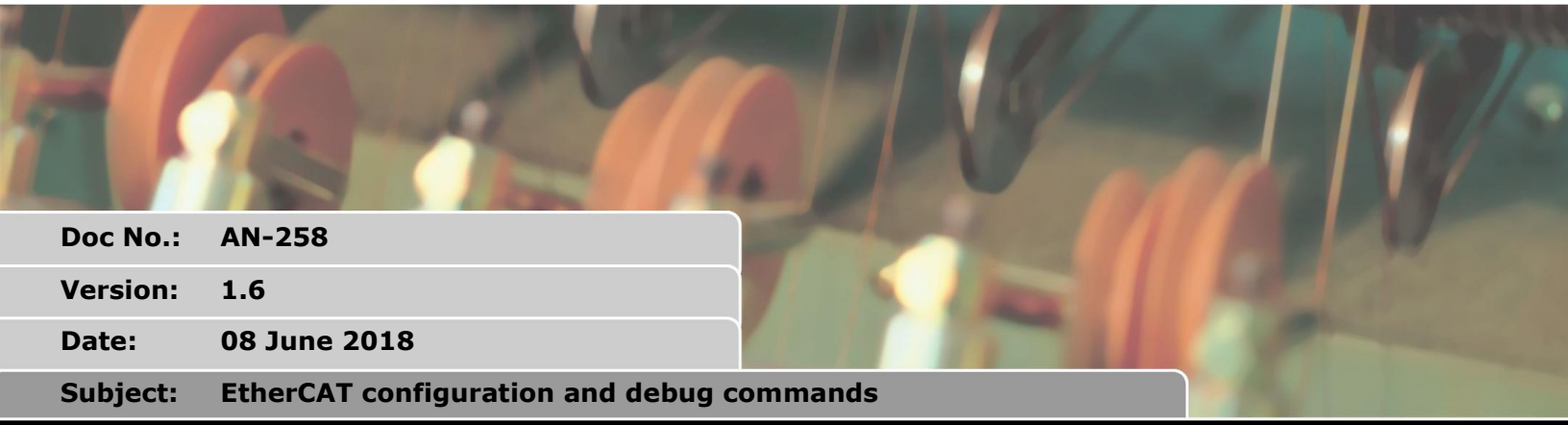
Trio Motion Technology Ltd.
Shannon Way, Tewkesbury,
Gloucestershire. GL20 8ND
United Kingdom
Tel: +44 (0)1684 292 333
Fax: +44 (0)1684 297 929

1000 Gamma Drive
Suite 206
Pittsburgh, PA 15238
United States of America
Tel: +1 412 968 9744
Fax: +1 412 968 9746

B1602 Tomson Centre
188 Zhang Yang Rd.,
Pudong New Area, Shanghai,
Postal code: 200122
CHINA
Tel/Fax: +86 21 587 97659



Doc No.: AN-258

Version: 1.6

Date: 08 June 2018

Subject: EtherCAT configuration and debug commands

APPLICATION NOTE

1. EtherCAT Master

This document covers the software used for EtherCAT initialisation. A typical system includes the Motion Coordinator, an EtherCAT module if applicable and a number of EtherCAT drives and IO modules. The MC4 range Technical Reference manual has information about the hardware and connections.

2. Version history

1.4	14th August 2014	Updated startup message format.
1.5	17th February 2015	Added ESC error code table
1.6	8th July 2018	Added ESC errorcode 0x03 description

Contents

3. Introduction

When the Motion Coordinator is powered up, the system software automatically searches for and initialises the EtherCAT servodrives and I/O devices connected to it. The initialisation is started on power up but can also be started under program control.

This document covers the initialisation process and also has descriptions of some of the more useful commands that can be used with the Motion Coordinator and EtherCAT. Commands can be used for starting and stopping the EtherCAT protocol, setting some optional parameters, interrogating the EtherCAT status and reading/writing CoE objects in the remote slaves.

Commands included in this document:

```
AUTO_ETHERCAT
AXIS_ENABLE
CO_READ
CO_READ_AXIS
CO_WRITE
CO_WRITE_AXIS
DRIVE_MODE
DRIVE_PROFILE
ETHERCAT
WDOG
```

The functions are as found in MC4N-ECAT/MC464 system software V2.0240.

4. System Initialisation

On power-up, the Motion Coordinator automatically starts the EtherCAT initialisation. It uses a stored table of known EtherCAT devices. When an EtherCAT node is discovered, the product code for that device is read from the node and compared with the known product codes in the table. When a match is found, the device is initialised and added to the network. If a match cannot be found then that device cannot be connected to the master and the initialisation stops.

If you have a new device which needs connecting, please contact Trio to have it added to the product table.

4.1. Startup messages

Terminal 0 shows the following messages while the EtherCAT is starting.

```
EtherCAT State: Init (0)
EtherCAT State: Pre-Operational (0)
EtherCAT State: Safe-Operational (0)
EtherCAT State: Operational (0)
EtherCAT Configuration (0):
    DRIVENAME   :  0 : 0 : 1 (1:0:0)
    DRIVENAME   :  1 : 0 : 2 (2:0:0)
    DRIVENAME   :  2 : 0 : 3 (3:0:0)
    DRIVENAME   :  3 : 0 : 4 (4:0:0)
```

4.2. EtherCAT states

Initial state: System is discovering devices and setting up the distributed clock parameters.

Pre-Operational System is configuring the PDO messages, operation mode and timing

state: parameters in each device.

Safe-Operational state: Network is running cyclically and is ready for full operation. Drives cannot be enabled in this state but position information can be seen in the Motion Coordinator axes.

Operational state: Network is fully operational. Cyclic messages are being transferred and the drives can be enabled via the control word.

4.3. EtherCAT configuration report

The EtherCAT configuration is shown after the network goes into Operational state.

```
DRIVENAME    :  A : B : C (D:E:F)
```

DRIVENAME: Shows a short name of the connected servo drive.

A: The node location on the network. 0 = first node, 1 = second node etc.

B: The Node Address set by switches or an EPROM parameter in the node.

C: The Node Address assigned by the EtherCAT master.

D: The Axis number assigned to the drive by the master.

E: The Profile number in use. (Set by DRIVE_PROFILE)

F: The axis Mode in use. (Set by DRIVE_MODE)

4.4. Setting the Node Address

If the Node Address switches or the Node Address parameter in the servo drive is set to 0, then the master allocates a Node Address automatically according to the position of the device on the cable network. For example, 3 servodrives are connected and they all have their Node Address set to 0.

```
DRIVENAME    :  0 : 0 : 1 (0:0:0)
DRIVENAME    :  1 : 0 : 2 (1:0:0)
DRIVENAME    :  2 : 0 : 3 (2:0:0)
```

If the Node Address switches, or Node Address parameters, are set to non-zero, then the master will allocate Node and Axis numbers according to the set addresses.

```
DRIVENAME    :  0 : 5 : 5 (4:0:0)
DRIVENAME    :  1 : 6 : 6 (5:0:0)
DRIVENAME    :  2 : 7 : 7 (6:0:0)
```

The slave must support ESC register 0x12 and place the Configured Address in that register for the master to read it.

4.5. Axis allocation rules

Axis numbers are set automatically using the Node Address and the AXIS_OFFSET parameter. Each Node is allocated an axis number according to the following equation:

Axis_number = AXIS_OFFSET + Node Address - 1

For example, AXIS_OFFSET SLOT(0) is set to 6. There are 4 EtherCAT servo drives connected with their Node Address switches set to 0.

```
DRIVENAME    :  0 : 0 : 1 (6:0:0)
DRIVENAME    :  1 : 0 : 2 (7:0:0)
DRIVENAME    :  2 : 0 : 3 (8:0:0)
```

```
DRIVENAME    :  3 : 0 : 4 (9:0:0)
```

5. Enable and disable

Before axes can be enabled, the EtherCAT network must be in Operational state. All axes are enabled together when the WDOG=ON command is issued.

WDOG=ON : Sets the CoE control word (0x6040) to the correct value for switch-on operation.

WDOG=OFF : Sets the CoE control word (0x6040) to the correct value for switch-off/disable.

Object 0x6040:00 is part of the cyclic PDO data and it is automatically set to the hexadecimal value 0x06 when WDOG is OFF. The value for WDOG ON depends on the operating mode but is usually 0x0F, 0x1F or 0x2F.

5.1. Axis_Enable

Individual axes can be disabled and enabled by using the axis parameter AXIS_ENABLE. **AXIS_ENABLE sets the control word 0x6040 for one axis only, leaving the other axes' control words enabled.**

WDOG must be ON for any axis to be enabled.

Axis n is enabled only when WDOG=ON AND AXIS_ENABLE AXIS(n)=1

Example 1: Disable axis 3 but leave all other axes enabled.

```
AXIS_ENABLE AXIS(3)=0
```

Example 2: Re-enable axes 10 and 11.

```
AXIS_ENABLE AXIS(10)=1
AXIS_ENABLE AXIS(11)=1
```

6. Starting and stopping the EtherCAT operation

The system software automatically starts the EtherCAT when it finds suitable slave devices on the network. If the EtherCAT protocol fails to start because of a problem, the following commands can be used to find the EtherCAT status, restart the EtherCAT initialisation and stop the EtherCAT protocol operation.

6.1. Start or re-start the EtherCAT initialisation

```
ETHERCAT(0, slot_number)

ETHERCAT(0,0) ' typical command in the MC4N-ECAT
```

6.2. Read EtherCAT status

Print the EtherCAT status to the terminal.

```
ETHERCAT(6, slot_number, -1)
```

Put the EtherCAT status in a VR variable.

```
ETHERCAT(6, slot_number, 20) ' result is put in VR(20)
IF VR(20)=1 THEN
  PRINT "EtherCAT in Pre-Op state"
ELSEIF VR(20)=2 THEN
  PRINT "EtherCAT in Safe-Op state"
ELSEIF VR(20)=3 THEN
```

```
  PRINT "EtherCAT in Op state"
ENDIF
```

6.3. Stop the EtherCAT protocol

```
ETHERCAT(1, slot_number)
```

6.4. See the number of Axes found

```
ETHERCAT(3, slot_number, -1)
```

7. MC_CONFIG settings

These parameters must be placed in the special MC_CONFIG program. They will have no function if used in the command line terminal or if placed in any other program.

7.1. Disable automatic EtherCAT initialization

```
AUTO_ETHERCAT = OFF 'do not start the EtherCAT network on power-up
```

After power-up, the EtherCAT can then be started by the user's own program using the ETHERCAT commands.

7.2. Set the axis to SPEED or TORQUE mode

```
DRIVE_MODE AXIS(2) = 1 ' axis 2 will start in position mode (default)
```

When in position mode, the ATYPE is automatically set to 65. The position loop gains in the drive must be set. The P_GAIN, I_GAIN etc. do nothing when the drive is in position control mode. The DAC_OUT value is the raw position command to the drive.

```
DRIVE_MODE AXIS(2) = 2 ' axis 2 will start in speed mode
```

When in speed mode, the ATYPE is automatically set to 66. The P_GAIN, I_GAIN etc. must be set in the Motion Coordinator for position loop control. The DAC_OUT value is proportional to velocity.

```
DRIVE_MODE AXIS(2) = 3 ' axis 2 will start in torque mode
```

When in torque mode, the ATYPE is automatically set to 67. The P_GAIN, I_GAIN etc. must be set in the Motion Coordinator for position loop control. The DAC_OUT value is proportional to Torque.

7.3. Set the axis to use an alternative EtherCAT profile

```
DRIVE_PROFILE AXIS(1) = 2 ' axis 1 will use EtherCAT profile 2
```

The profiles are contained in EtherCAT slave device lookup table. The exact function of the profile will be different for each slave device and for each mode of operation. See the profile tables in the Motion Coordinator for details.

One example of an alternative profile is to make the DRIVE_FE and Touch Trigger active. Touch Trigger is the EtherCAT mechanism that links to the Trio REGIST function.

7.4. MC_CONFIG example

Set up 4 drives to use speed mode. (cyclic synchronous velocity)

2 axes will use the default profile, sending target speed and control word.
2 axes will use an extended mode which also has the Touch Probe data in the cyclic telegram so that the REGIST command can be used.

```
' Set up 2 axes for default speed mode
DRIVE_MODE AXIS(0)=2
DRIVE_PROFILE AXIS(0)=0
DRIVE_MODE AXIS(1)=2
DRIVE_PROFILE AXIS(1)=0

' Set up axes 2 and 3 to use drive registration
DRIVE_MODE AXIS(2)=2
DRIVE_PROFILE AXIS(2)=2
DRIVE_MODE AXIS(3)=2
DRIVE_PROFILE AXIS(3)=2
```

8. Read CoE objects from an EtherCAT slave device

CoE means CanOpen-over-EtherCAT. The EtherCAT system must be, as a minimum, in Pre-Op state for the CanOpen object read command to work. To find the state, use the ETHERCAT command function 6 as described in section 2.1.

8.1. CoE object read

```
CO_READ(slot_number, address, index, subindex ,type [,output])
```

Slot_number:	The slot number of the master EtherCAT device. (Usually slot 0 but varies on the MC664(X))
Address:	Configured EtherCAT address of node to be accessed.
Index:	CanOpen object index.
SubIndex:	CanOpen object subindex.
Type:	Data type of the object (see CanOpen Object Type Table)
Optional Output:	Pointer to VR array to read data into or -1 if data to be returned to the current output device. (Terminal 0) Default -1.

8.1.1. CO_READ Example

Print the value for object 0x60C2 sub-index 01, position interpolation time units. This is an unsigned 8 bit value and so has the CanOpen type 0x05. Value is to be read from servo drive at EtherCAT configured address 100.

```
CO_READ(slot_number, 100, $60c2, 1, 5, -1)
```

Print the value for object 0x60C2 sub-index 02, position interpolation time index. This is an 8 bit value and so has the CanOpen type 0x02. Value is to be read from servo drive at EtherCAT configured address 100.

```
CO_READ(slot_number, 100, $60c2, 2, 2, -1)
```

In a system running at 1 msec servo period these objects should return 1 and -3 respectively.

8.2. CoE axis object read

```
CO_READ_AXIS(axis_number, index, subindex ,type [,output])
```

Axis:	Configured axis number for the node to be addressed.
Index:	CanOpen object index.
SubIndex:	CanOpen object subindex.
Type:	Data type of the object (see CanOpen Object Type Table)
Optional Output:	Pointer to VR array to read data into or -1 if data to be returned to the current output device. (Terminal 0) Default -1.

8.2.1. CO_READ_AXIS Example

Print the value for object 0x6064 sub-index 00, position actual value. This is a 32 bit long word and so has the CanOpen type 0x04. Value is to be read from servo drive at axis 5.

```
CO_READ_AXIS(5, $6064, 0, 4, -1)
```

This shows the position value from the drive and it should change as the motor shaft turns. Normally this value will be the same as the parameter ENCODER for that axis.

9. Write CoE objects to an EtherCAT slave device

The EtherCAT system must be, as a minimum, in Pre-Op state for the CanOpen object read command to work. To find the state, use the ETHERCAT function 6 command as described in section 2.1.

9.1. CoE object write

```
CO_WRITE(slot_number, address, index, subindex ,type ,data [,value])
```

Slot_number:	The slot number of the master EtherCAT device. (Usually slot 0 but varies on MC664(X))
Address:	Configured EtherCAT address of node to be accessed.
Index:	CanOpen object index.
SubIndex:	CanOpen object subindex.
Type:	Data type of the object. (see CanOpen Object Type Table)
Data:	0 to n specifies the VR or the start of the VR block containing the data to send. -1 specifies that the value is in the next parameter [Value].
Optional Value:	The data value to be sent to remote node/drive.

9.1.1. CO_WRITE example

Write a position controller proportional gain value to the servo drive at configured EtherCAT address 104. CoE object 0x60FB sub-index 0x01, type 6 (unsigned 16 bit integer).

```
CO_WRITE(slot_number, 104, $60fb, 1, 6, -1, 2500)
```

Write a position controller velocity feedforward gain value to the servo drive at configured EtherCAT address 102. CoE object 0x60FB sub-index 0x02, type 6 (unsigned 16 bit integer).

```
VR(35)=1000
' write the value from VR(35)
CO_WRITE(slot_number, 102, $60fb, 2, 6, 35)
```

Warning: Always refer to the manufacturer's user manual before writing to a CanOpen object over

EtherCAT.

9.2. CoE axis object write

```
CO_WRITE_AXIS(axis_number, index, subindex ,type ,data [,value])
```

Axis_number:	Configured axis number for the node to be addressed.
Index:	CanOpen object index.
SubIndex:	CanOpen object subindex.
Type:	Data type of the object. (see CanOpen Object Type Table)
Data:	0 to n specifies the VR or the start of the VR block containing the data to send. -1 specifies that the value is in the next parameter [Value].
Optional Value:	The data value to be sent to remote node/drive.

9.2.1. CO_WRITE_AXIS example

Write a value of 1 to a manufacturer specific object on servo drive at axis 3. CoE object 0x2802 sub-index 0x00, type 2 (8 bit integer).

```
CO_WRITE_AXIS(3, $2802, 0, 2, -1, 1)
```

Write a position controller velocity feedforward gain value to the servo drive at axis 12. CoE object 0x60FB sub-index 0x02, type 6 (unsigned 16 bit integer).

```
VR(2010)=1000
' write the value from VR(2010)
CO_WRITE_AXIS(axis_number, $60fb, 2, 6, 2010)
```

Warning: Always refer to the manufacturer's user manual before writing to a CanOpen object over EtherCAT.

10. CanOpen Object Type Table

For use with CO_READ, CO_READ_AXIS, CO_WRITE and CO_WRITE_AXIS "type" parameter.

- Object Dictionary Data Types (CiA DS301)

0x01	Boolean
0x02	Integer8
0x03	Integer16
0x04	Integer32
0x05	Unsigned8
0x06	Unsigned16
0x07	Unsigned32

11. ETHERCAT command syntax

The ethercat commands all follow the format :

```
ETHERCAT(function, param1 [,param2 …])
```

Here is a list of some useful function numbers.

```
INIT_PROTOCOL       = $00,
CLOSE_PROTOCOL      = $01,
GET_MASTER_STATE    = $06,
```

```
GET_MASTER_ERR_STATUS = $07,

GET_RING_STATE     = $22,    /* return current master state.*/
RESET_SLAVE_ERROR  = $64,    /* reset a drive error */
NW_DISPLAY_CONFIG  = $87,    /* Display configuration to terminal */
```

11.1. Example GET_MASTER_STATE

```
ETHERCAT($06, slot_number, vr_index)
```

Motion Coordinator's Master state is returned in VR(vr_index)

State value => 0 : init, 1 : pre-op, 2 : safeop, 3 : op

11.2. Example GET_MASTER_ERR_STATUS

```
ETHERCAT($07, slot_number, vr_index)
```

Latest master error code is returned in VR(vr_index)

11.3. Example RESET_SLAVE_ERROR

```
ETHERCAT($64, slave_axis [,mode [,timeout_value]])
```

Cycles the Reset bit of the control word to the drive which is configured for the selected axis.

Parameters

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

Notes:

Note that this function initiates the fault reset and then returns, it does not wait for the status word's fault flag to clear or for the timeout period. Also its true/false response is based upon whether it successfully initiated the fault reset, not whether the error was cleared.

Prior to firmware release v2.0195 this function would only run mode 0 and therefore set the 'error reset' in the drive control word for 8 servo periods, and the drive status 'Fault' flag (bit 3) was not checked.

11.4. Example NW_DISPLAY_CONFIG

Displays network configuration to the terminal (Terminal 0)

```
ETHERCAT($87, slot_number)
```

Response, servos:

```
DRIVENAME   node : Address_Switch : Address_allocated (Axis:prof:mode)
```

Response, IO devices, and GRT coupler:

```
vendor : product: configured address ( start addr : number digital
inputs/start addr : number digital inputs [/ start addr : number analog
inputs / start addr : number analog outputs]
```

11.5. Printing ETHERCAT results to the terminal

If the ETHERCAT command is typed in the terminal, it is better to see the result displayed directly in the terminal. Any ETHERCAT command which has a vr_index parameter can do this. Setting vr_index to -1 will display the value on the terminal and not put it in a VR.

ETHERCAT commands return TRUE (-1) or FALSE (0) to indicate the success of the command. If a command does not seem to function correctly put PRINT before the command and see the value returned. For example:

```
PRINT ETHERCAT(6, slot_number, 20)
>>-1.0000
```

-1.0000 indicates that the command was executed successfully.

12. Checking EtherCAT configuration details

The following sub-set of functions can be used to discover the EtherCAT slaves supported and their specific configuration details.

```
GET_SLAVE_CONFIG = $104, /* Display internal slave configurations */
GET_SLAVE_IO_CONFIG = $108, /* Display slave IO configuration */
GET_SLAVE_SM_CONFIG = $10C, /* Display slave Sync Manager config. */
GET_SLAVE_COE_CONFIG = $111, /* Display slave CoE configuration */
GET_SLAVE_PDO_CONFIG = $116, /* Display slave PDO configuration */
GET_SLAVE_CYC_CONFIG = $11A, /* Display slave Cyclic configuration */
```

12.1. Example GET_SLAVE_CONFIG

Displays a list of the internally held slave configurations to the terminal (Terminal 0)

Syntax:

```
ETHERCAT($104[, VendorID[, ProductCode]])
```

Response:

```
Manufacturer name  (VendorID), Drive name  (Product Code), ATYPE
```

Example 1:

```
>>ETHERCAT($104)
```

Response:

```
ABB             (0x000000B7), ACS355         (0x000001F7), 68
BAUMER          (0x000000EC), Encoder 2byte  (0x0000001E), 69
CT              (0x000000F9), UNIDRIVE SP    (0x00040003), 65
STOBER          (0x000000B9), POSIDRIVE      (0x00001388), 66
```

Example 2:

```
>>ethercat($104,$509)
```

Response:

```
TR-ELECTRONIC  (0x00000509), LINEAR ENC    (0x04D2AB01), 69
TR-ELECTRONIC  (0x00000509), LINEAR ENC    (0x303AAE0B), 69
TR-ELECTRONIC  (0x00000509), ROTARY ENC    (0x0517586A), 69
TR-ELECTRONIC  (0x00000509), ROTARY ENC    (0x05175840), 69
TR-ELECTRONIC  (0x00000509), ROTARY ENC    (0x0515BAC7), 69
TR-ELECTRONIC  (0x00000509), ROTARY ENC    (0x03430701), 69
```

12.2. Example GET_SLAVE_IO_CONFIG

Displays the internally held input/output slave configurations to the terminal (Terminal 0)

Syntax:

```
ETHERCAT($108[, VendorID[, ProductCode]])
```

Response:

```
Manufacturer name  (VendorID), Drive name  (Product Code), ATYPE
Number of axes on this device,
Number of digital outputs,
Number of digital inputs,
Number of analogue outputs,
Number of analogue inputs.
```

If there is more than one entry for a particular drive, then the first one is the default. The others are accessed by setting DRIVE_MODE and DRIVE_PROFILE in the MC_CONFIG file.

Example 1:

```
>>ETHERCAT($108,$F9,$40203)
```

Response:

```
CT             (0x000000F9), DIGITAX       (0x00040203), 65
Num of Axes: 1, Dig Out: 0, Dig In: 0, Analogue Out: 0, Analogue In: 0
CT             (0x000000F9), DIGITAX       (0x00040203), 65
Num of Axes: 2, Dig Out: 0, Dig In: 0, Analogue Out: 0, Analogue In: 0
CT             (0x000000F9), DIGITAX       (0x00040203), 65
Num of Axes: 1, Dig Out: 0, Dig In: 16, Analogue Out: 0, Analogue In: 0
CT             (0x000000F9), DIGITAX       (0x00040203), 66
Num of Axes: 1, Dig Out: 0, Dig In: 0, Analogue Out: 0, Analogue In: 0
CT             (0x000000F9), DIGITAX       (0x00040203), 66
Num of Axes: 2, Dig Out: 0, Dig In: 0, Analogue Out: 0, Analogue In: 0
CT             (0x000000F9), DIGITAX       (0x00040203), 67
Num of Axes: 1, Dig Out: 0, Dig In: 0, Analogue Out: 0, Analogue In: 0
```

In the above list, the entries are:

Entry	Description	MC_CONFIG setting
1	Position mode, one axis. (Default)	DRIVE_MODE=1
		DRIVE_PROFILE=1
2	Position mode with auxiliary encoder	DRIVE_MODE=1
		DRIVE_PROFILE=2
3	Position mode with one axis and 16 digital inputs	DRIVE_MODE=1
		DRIVE_PROFILE=3

4	Speed mode with one axis	DRIVE_MODE=2 DRIVE_PROFILE=1
5	Speed mode with auxiliary encoder	DRIVE_MODE=2 DRIVE_PROFILE=2
6	Torque mode with one axis	DRIVE_MODE=3 DRIVE_PROFILE=1

12.3. Example GET_SLAVE_SM_CONFIG

Displays the internally held Sync Manager slave configurations to the terminal (Terminal 0)

Syntax:

```
ETHERCAT($10C[, VendorID[, ProductCode[, DriveMode]])
```

Response:

```
Manufacturer name  (VendorID), Drive name  (Product Code), ATYPE
Index Number,
Enable state,
CoE Object Address,
Length in bytes,
Control Code,
Trio internal function number.
```

Example 1:

```
>>ETHERCAT($10C,$6A,$414B44,2)
```

Response:

```
KOLLMORGEN     (0x0000006A), AKD           (0x00414B44), 66
Index: 0, Enable: 1, Addr: 0x00001800, Length: 128, Ctrl: 0x00000026, Func: 1
Index: 1, Enable: 1, Addr: 0x00001C00, Length: 128, Ctrl: 0x00000022, Func: 2
Index: 2, Enable: 1, Addr: 0x00001100, Length: 6, Ctrl: 0x00000024, Func: 3
Index: 3, Enable: 1, Addr: 0x00001140, Length: 6, Ctrl: 0x00000020, Func: 4
>>
```

12.4. Example GET_SLAVE_COE_CONFIG

Displays the internally held CanOpen over EtherCAT slave configurations to the terminal. (Terminal 0) The configuration data is sent to the drive during the startup sequence and is used by the drive to define its CoE interface.

Syntax:

```
ETHERCAT($111[, VendorID[, ProductCode[, DriveMode]])
```

Response:

```
Manufacturer name  (VendorID), Drive name  (Product Code), ATYPE
Object Index:Sub Index (?) = value on state change
```

Example 1:

```
>>ETHERCAT($111,$539,$2200001,1)
```

Response:

```
YASKAWA        (0x00000539), SGDV          (0x02200001), 65
0x1C12:0x00 ( 1) = 0x00000000 on Pre-op to Safe-op
0x1C13:0x00 ( 1) = 0x00000000 on Pre-op to Safe-op
0x1A01:0x00 ( 1) = 0x00000000 on Pre-op to Safe-op
0x1A01:0x01 ( 4) = 0x60410010 on Pre-op to Safe-op
0x1A01:0x02 ( 4) = 0x60640020 on Pre-op to Safe-op
0x1A01:0x00 ( 1) = 0x00000002 on Pre-op to Safe-op
0x1601:0x00 ( 1) = 0x00000000 on Pre-op to Safe-op
0x1601:0x01 ( 4) = 0x60400010 on Pre-op to Safe-op
0x1601:0x02 ( 4) = 0x607A0020 on Pre-op to Safe-op
0x1601:0x00 ( 1) = 0x00000002 on Pre-op to Safe-op
0x1C12:0x01 ( 2) = 0x00001601 on Pre-op to Safe-op
0x1C12:0x00 ( 1) = 0x00000001 on Pre-op to Safe-op
0x1C13:0x01 ( 2) = 0x00001A01 on Pre-op to Safe-op
0x1C13:0x00 ( 1) = 0x00000001 on Pre-op to Safe-op
0x6060:0x00 ( 1) = 0x00000008 on Pre-op to Safe-op
0x60C2:0x01 ( 1) = 0x00000001 on Pre-op to Safe-op
0x60C2:0x02 ( 1) = 0x000000FD on Pre-op to Safe-op
>>
```

12.5. Example SLAVE_PDO_CONFIG

Displays the internally held slave PDO data structures to the terminal. (Terminal 0)

Syntax:

```
ETHERCAT($116[, VendorID[, ProductCode[, DriveMode]])
```

Response:

```
Manufacturer name  (VendorID), Drive name  (Product Code), ATYPE
Cyclic output structure (ie data from drive to master)
Cyclic input structure (ie data from master to drive)
```

Example 1:

```
>>ETHERCAT($116,$16,$20001,2)
```

Response:

```
LTi DRiVES     (0x00000016), ServoOne      (0x00020001), 66
Cyclic data objects output from the slave
0: Status Word (2 bytes)
1: Actual Position (4 bytes)
Cyclic data objects input to slave
0: Control Word (2 bytes)
1: Target Speed (4 bytes)
>>
```

12.6. Example SLAVE_CYC_CONFIG

Displays the internally held cyclic operation data for the slaves to the terminal. (Terminal 0)

Syntax:

```
ETHERCAT($11A[, VendorID[, ProductCode[, DriveMode]])
```

Response:

```
Manufacturer name  (VendorID), Drive name  (Product Code), ATYPE
```

```
Control mode, Type, States, Command, Length
```

Example 1:

```
>>ETHERCAT($11A, $557)
```

Response:

```
Jenny Science AG (0x00000557), Xvi (0x00007508), 65, (0)
Ctrl Mode: 8, UseLrdLwr: 0, States: 0x000C, Length: 6
Jenny Science AG (0x00000557), Xvi (0x00007508), 65, (1)
Ctrl Mode: 8, UseLrdLwr: 0, States: 0x000C, Length: 14
>>
```

13. Special considerations

13.1. Delayed startup

In some cases a drive requires additional parameters in the startup command as it takes longer to "discover" this drive on the network. It is unlikely that such a drive will start automatically when the Motion Coordinator powers up, so it will be necessary to start it manually in a BASIC program.

The following program shows a method for starting a network with servo drives requiring delayed start. Note that the ETHERCAT command which starts the network takes some time to execute and so the program must wait for the EtherCAT network to arrive in Operational state before it continues to initialize other axis parameters.

Syntax: ETHERCAT(Fn, slot [,MAC_retries])

Program

```
ETHERCAT(0,0,4)
REPEAT
  ETHERCAT(6, 0, 20) ' result is put in VR(20)
UNTIL VR(20)=3
PRINT "EtherCAT Initialization complete"

axis_number=0
counts_per_rev=4096
max_motor_speed=3000 'SPEED IN rpm
BASE(axis_number)
UNITS=1
DEFPOS(0)
SPEED=5000
ACCEL=500000
DECEL=500000
SERVO=ON
WDOG=ON
```

Results in terminal 0

```
EtherCAT State: Init (0)
EtherCAT State: Pre-Operational (0)
EtherCAT State: Safe-Operational (0)
EtherCAT State: Operational (0)
EtherCAT Configuration (0):
    DRIVENAME  :  0 : 1 : 1 (0:0:0)
EtherCAT Initialization complete
```

14. ESC Error codes

The ESC, or EtherCAT Slave Controller is the part of the slave device that is responsible for low-level operation of the EtherCAT communications. During startup, if there is a problem, the ESC can report an error code to the master. For example in the Motion Perfect terminal:

```
EtherCAT State: Init (0)
EtherCAT Configuration (0):
Drivename: 0 : 0 : 1 (0:0:0) - ESC Status: 0x11, Code: 0x14
```

14.1. Error Code table

Code	Description	Current state (or state change)	Resulting state
0x0000	No error	Any Current state	
0x0001	Unspecified error	Any	Any + E
0x0002	No memory	Any	Any + E
0x0003	Invalid Device Setup	Any	Any + E
0x0011	Invalid requested state change	I→S, I→O, P→O, O→B, S→B, P→B	Current state + E
0x0012	Unknown requested state	Any	Current state + E
0x0013	Bootstrap not supported	I→B	I + E
0x0014	No valid firmware	I→P	I + E
0x0015	Invalid mailbox configuration	I→B	I + E
0x0016	Invalid mailbox configuration	I→P	I + E
0x0017	Invalid sync manager configuration	P→S, S→O	Current state + E
0x0018	No valid inputs available	O, S, P→S	P + E
0x0019	No valid outputs	O, S→O	S + E
0x001A	Synchronization error	O, S→O	S + E
0x001B	Sync manager watchdog	O, S	S + E
0x001C	Invalid Sync Manager Types	O, S P→S S + E	P + E
0x001D	Invalid Output Configuration	O, S P→S S + E	P + E
0x001E	Invalid Input Configuration	O, S, P→S	P + E
0x001F	Invalid Watchdog Configuration	O, S, P→S	P + E
0x0020	Slave needs cold start	Any	Current state + E
0x0021	Slave needs INIT	B, P, S, O	Current state + E

0x0022	Slave needs PREOP	S, O	S + E, O + E
0x0023	Slave needs SAFEOP	O	O + E
0x0024	Invalid input mapping	P→S	P + E
0x0025	Invalid output mapping	P→S	P + E
0x0026	Inconsistent settings	P→S	P + E
0x0027	Free-Run not supported	P→S	P + E
0x0028	Synchronization not supported	P→S	P + E
0x0029	Free-Run needs 3 buffer mode	P→S	P + E
0x002A	Background watchdog	S, O	P + E
0x002B	No valid inputs and outputs	O, S→O	S + E
0x002C	Fatal Sync error	O	S + E
0x002D	No Sync error	S→O	S + E
0x0030	Invalid DC SYNCH Configuration	O, S	S + E
0x0031	Invalid DC Latch Configuration	O, S	S + E
0x0032	PLL Error	O, S	S + E
0x0033	Invalid DC IO Error	O, S	S + E
0x0034	Invalid DC Timeout Error	O, S	S + E
0x0035	DC invalid Sync Cycle Time	P→S	P + E
0x0036	DC Sync0 Cycle Time	P→S	P + E
0x0037	DC Sync1 Cycle Time	P→S	P + E
0x0041	MBX_AOE	B, P, S, O	Current state + E
0x0042	MBX_EOE	B, P, S, O	Current state + E
0x0043	MBX_COE	B, P, S, O	Current state + E
0x0044	MBX_FOE	B, P, S, O	Current state + E
0x0045	MBX_SOE	B, P, S, O	Current state + E
0x004F	MBX_VOE	B, P, S, O	Current state + E
0x0050	EEPROM no access	Any	Any + E
0x0051	EEPROM error	Any	Any + E
0x0060	Slave restarted locally	Any	I
Other codes			
<0x8000	Reserved		
0x8000-0xFFFF	Vendor specific		

I = Initial State
P = Pre-Operational
S = Safe-Operational
O = Operational
B = Boot State

E = Error bit set in Status byte

15. Glossary

CoE CanOpen over EtherCAT

CO CanOpen. A protocol layer developed originally for use over CAN bus.

Object A parameter, control or similar value in the remote slave device. CanOpen objects are addressed using a 16 bit Index and an 8 bit sub-index. E.g. 0x6061:00 is the mode of operation in a remote drive.

0x Indicates that a number is to be shown in Hexadecimal format.

$ Prefix to a hexadecimal number in Trio BASIC.