PSI
TCP-IP to CANopen Gateway documentation

Author Matthias Kamber / Sven Schlienger
Date 01.12.2025
Revision 2.3

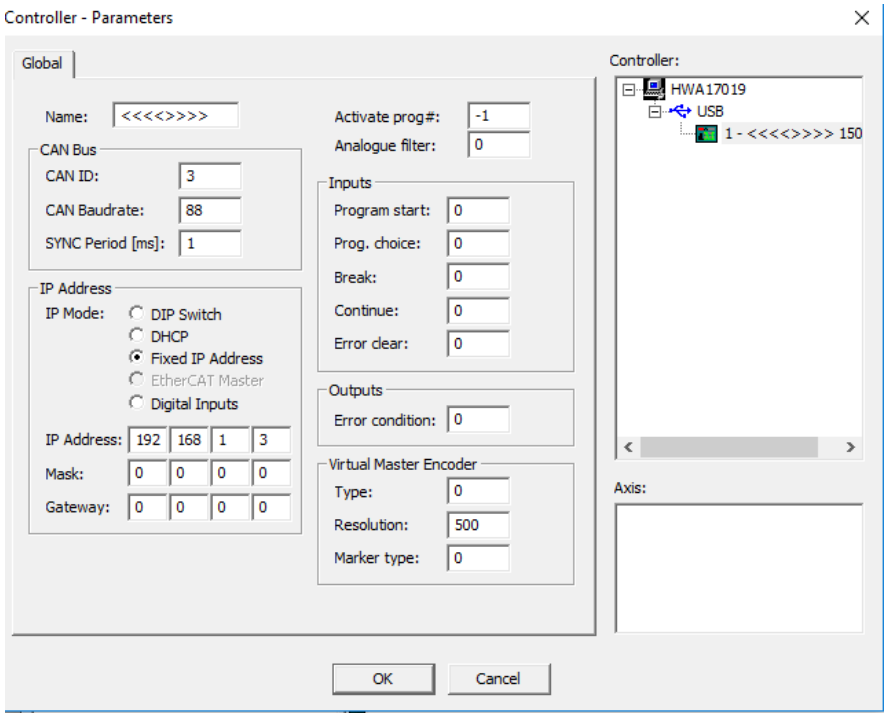
History:
1.0 Initial version
1.1 Change of UserUnits factor
1.2 Some Units updated in list
1.3 New units for “Axis Inversion” and “Slave Axis direction” (1=normal / -1=inverted)
1.4 Device code “C_DC_COBLEY” added. New command (16, 17, 80, 81, 82, 83, 84, 85) added.
1.5 New command (56, 91) added. “Control Brake” feature
1.6 New command (86, 33, 34) added. Limits in UserUnits.
1.7 Description for integrating an CANopen IO module.
1.8 Internal definitions for PSI, improvements to the communication process, detailed description of individual commands, add command (60, 99)
1.9 Changes for the new TCP communication without header and ETX
2.1 On Enable-> Write Parameter
2.2 New Inhibit explanation.
2.3 Velocity-mode implementation, new commands(InPositionBand, ActualVelocity, DynamicLimits and Possible ModesOfOperation

1 Introduction

This documentation covers the interface and some information for the TCP-IP to CANopen gateway application.

1.1 Set IP address and CAN baudrate

Connect Aposs-IDE to the MasterMACS and choose <Controller> -> <Parameters> -> <Global / Axis>. In the block “IP Address” the required settings can be done. The CAN baudrate can be set in the block “CAN Bus”. The lower digit represents CAN1 and the higher CAN2. The baudrate is given with a number between 0 and 8 (see below).



Number	Baudrate	Used in PSI
0	5 kBaud	-
1	10 kBaud	-
2	20 kBaud	-
3	50 kBaud	-
4	100 kBaud	-
5	125 kBaud	YES
6	250 kBaud	YES
7	500 kBaud	YES
8	1000 kBaud	YES

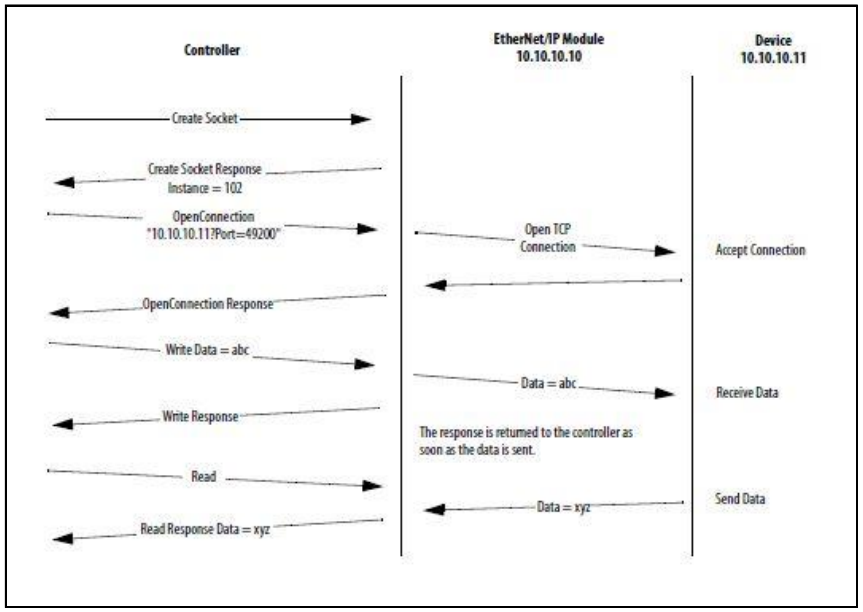
2 TCP communication

2.1 Open TCP Port

For open the TCP socket a client must open it with the standard TCP handshake with the correct IP address and port number. The MasterMACS/MicroMACS6 acts as TCP server. The two TCP Server are open in the application using the command “EthernetOpenServer(…)” MasterMACS/MicroMACS6 supports the following port numbers. There are 2 port number available for the PSI TCP communication

TCP_PORT_NUM_1	**1912**	**NICOS / EPICS**
TCP_PORT_NUM_2	**1913**	**Elektronik-Maintenance**
	23	**Telnet / Aposs**

TCP communication is shown in the following picture.



2.2 Write Data to MACS

2.2.1 Transmit PDO over TCP

After a socket is open successfully the client can write a packet. The transmit packet must have the following structure in the pay load of the TCP telegram. Most socket driver add the correct header of the telegram by themselves. Inside the MasterMACS this telegram is received by the application and the data are stored in a specific array. The application checks for new incoming messages and parses the string inside the array.
See the sample of a transmit telegram below. Only the green part is the PDO data which holds the string with ASCII characters for this application. Note that the size of the telegram can be different depending on the command string. It is also possible to keep the size of the transmit telegram constant and fill in the rest after the <CR> with 0x00.

The command to write is always composed as follows:
Axis Number + “S” + Command + “=” + Value + CR

Number	writeBuffer	Value	Explanation
1	0	0x33	PDO data 1 "3" (Axis Number)
2	1	0x53	PDO data 2 "S" (0x53); work also with "s"(0x73)!
3	2	0x30	PDO data 3 "0" (Command)
4	3	0x34	PDO data 4 "2" (Command)
5	4	0x3D	PDO data 5 "="
6	5	0x31	PDO data 6 "1" (Value)
7	6	0x32	PDO data 7 "2" (Value)
8	7	0x0D	PDO data 9 <CR>

This example sets the target position of axis 3 to 12.

2.2.2 Receive PDO over TCP

The string that are replied by the application could look something like:

3 S 02<ACK><CR>
3 S 02<NAK><CR>
3 S 10<CAN><CR> (STATUS is not writable)

Note that there is always a <CR> to signal the end of the string. The MasterMACS returns always data of 30 bytes. The expected response when setting the target position is shown below.
The MasterMACS repeats the command and indicates with the ACK or NAK whether it has been executed successfully. “CAN” indicates when a command is not valid.

The ACK/NAK is only relevant for a statement as to whether a driver is connected to the bus or not if the field in the “SDO (ACK/NAK)” column in section “5 Commands and Parameters” is colored.

Number	writeBuffer	Value	Explanation
1	0	0x33	PDO data 0 "3" (Axis Number)
2	1	0x20	PDO data 1 " " (space
3	2	0x53	PDO data 2 "S"
4	3	0x20	PDO data 3 " " (space)
5	4	0x32	PDO data 4 "2" (Command)
6	5	0x06	PDO data 5 "ACK" (0x06) ; “NAK” (0x15) ; “CAN” (0x18)
7	6	0x0D	PDO data 6 <CR>
8	7	0x00	PDO data 7 “NULL” (0x00)
9	8	0x00	PDO data 8 “NULL” (0x00)
10	9	0x00	PDO data 9 “NULL” (0x00)
11	10	0x00	PDO data 10 “NULL” (0x00)
12	11	0x00	PDO data 11 “NULL” (0x00)
13	12	0x00	PDO data 12 “NULL” (0x00)
14	13	0x00	PDO data 13 “NULL” (0x00)
15	14	0x00	PDO data 14 “NULL” (0x00)
16	15	0x00	PDO data 15 “NULL” (0x00)
17	16	0x00	PDO data 16 “NULL” (0x00)
18	17	0x00	PDO data 17 “NULL” (0x00)
19	18	0x00	PDO data 18 “NULL” (0x00)
20	19	0x00	PDO data 19 “NULL” (0x00)
21	20	0x00	PDO data 20 “NULL” (0x00)
22	21	0x00	PDO data 21 “NULL” (0x00)
23	22	0x00	PDO data 22 “NULL” (0x00)
24	23	0x00	PDO data 23 “NULL” (0x00)
25	24	0x00	PDO data 24 “NULL” (0x00)
26	25	0x00	PDO data 25 “NULL” (0x00)
27	26	0x00	PDO data 26 “NULL” (0x00)
28	27	0x00	PDO data 27 “NULL” (0x00)
29	28	0x00	PDO data 28 “NULL” (0x00)
30	29	0x00	PDO data 29 “NULL” (0x00)

2.3 Read Data from MasterMACS

2.3.1 Transmit PDO over TCP

Below you can see a telegram showing the read command. The green parts contain the PDOs with ASCII character for this application

The command to write is always composed as follows:
Axis Number + “R” + Command + CR

Number	writeBuffer	Value	Explanation
1	0	0x33	PDO data 1 "3" (Axis Number)
2	1	0x52	PDO data 2 "R" (0x52); work also with "r"(0x72)!
3	2	0x30	PDO data 3 "0" (Command)
4	3	0x32	PDO data 4 "2" (Command)
5	4	0x0D	PDO data 5 <CR>

This example read the target position of axis 3.

2.3.2 Receive PDO over TCP

The string that are replied by the application could look something like:

3 R 2=12.819<ACK><CR>
3 R 2<ACK><CR>
3 R 2<NAK><CR>
3 R 0<CAN><CR> (MOVE is not readable)

There are four ways in which a response to a read request can be validated. Either with a return value, an ACK, a NAK or a CAN (invalid). “CAN” indicates when a command is not permitted.

The ACK/NAK is only relevant for a statement as to whether a driver is connected to the bus or not if the field in the “SDO (ACK/NAK)” column in section “5 Commands and Parameters” is colored.

In this table you can see a request that returns a value. If a value can be read, this is equivalent to an ACK.

Number	writeBuffer	Value	Explanation
1	0	0x33	PDO data 0 "3" (Axis Number)
2	1	0x20	PDO data 1 " " (space
3	2	0x52	PDO data 2 "R"
4	3	0x20	PDO data 3 " " (space)
5	4	0x32	PDO data 4 "2" (Command)
6	5	0x3D	PDO data 5 "="
7	6	0x31	PDO data 6 “1”
8	7	0x32	PDO data 7 “2”
9	8	0x2E	PDO data 8 “.”
10	9	0x38	PDO data 9 “8”
11	10	0x06	PDO data 10 “ACK” (0x06)
12	11	0x0D	PDO data 11 <CR>
13	12	0x00	PDO data 12 “NULL” (0x00)
14	13	0x00	PDO data 13 “NULL” (0x00)
15	14	0x00	PDO data 14 “NULL” (0x00)
16	15	0x00	PDO data 15 “NULL” (0x00)
17	16	0x00	PDO data 16 “NULL” (0x00)
18	17	0x00	PDO data 17 “NULL” (0x00)
19	18	0x00	PDO data 18 “NULL” (0x00)
20	19	0x00	PDO data 19 “NULL” (0x00)
21	20	0x00	PDO data 20 “NULL” (0x00)
22	21	0x00	PDO data 21 “NULL” (0x00)
23	22	0x00	PDO data 22 “NULL” (0x00)
24	23	0x00	PDO data 23 “NULL” (0x00)
25	24	0x00	PDO data 24 “NULL” (0x00)
26	25	0x00	PDO data 25 “NULL” (0x00)
27	26	0x00	PDO data 26 “NULL” (0x00)
28	27	0x00	PDO data 27 “NULL” (0x00)
29	28	0x00	PDO data 28 “NULL” (0x00)
30	29	0x00	PDO data 29 “NULL” (0x00)

This table shows the response if no value is returned, a slave is not connected or a command is not permitted.

Number	writeBuffer	Value	Explanation
1	0	0x33	PDO data 0 "3" (Axis Number)
2	1	0x20	PDO data 1 " " (space
3	2	0x52	PDO data 2 "R"
4	3	0x20	PDO data 3 " " (space)
5	4	0x32	PDO data 4 "2" (Command)
6	5	0x06	PDO data 5 "ACK" (0x06) ; “NAK” (0x15) ; “CAN” (0x18)
7	6	0x0D	PDO data 6 <CR>
8	7	0x00	PDO data 7 “NULL” (0x00)
9	8	0x00	PDO data 8 “NULL” (0x00)
10	9	0x00	PDO data 9 “NULL” (0x00)
11	10	0x00	PDO data 10 “NULL” (0x00)
12	11	0x00	PDO data 11 “NULL” (0x00)
13	12	0x00	PDO data 12 “NULL” (0x00)
14	13	0x00	PDO data 13 “NULL” (0x00)
15	14	0x00	PDO data 14 “NULL” (0x00)
16	15	0x00	PDO data 15 “NULL” (0x00)
17	16	0x00	PDO data 16 “NULL” (0x00)
18	17	0x00	PDO data 17 “NULL” (0x00)
19	18	0x00	PDO data 18 “NULL” (0x00)
20	19	0x00	PDO data 19 “NULL” (0x00)
21	20	0x00	PDO data 20 “NULL” (0x00)
22	21	0x00	PDO data 21 “NULL” (0x00)
23	22	0x00	PDO data 22 “NULL” (0x00)
24	23	0x00	PDO data 23 “NULL” (0x00)
25	24	0x00	PDO data 24 “NULL” (0x00)
26	25	0x00	PDO data 25 “NULL” (0x00)
27	26	0x00	PDO data 26 “NULL” (0x00)
28	27	0x00	PDO data 27 “NULL” (0x00)
29	28	0x00	PDO data 28 “NULL” (0x00)
30	29	0x00	PDO data 29 “NULL” (0x00)

3 Internal Parameter for the system

In the application the axis settings and the axis parameters are stored in an array. For each axis there is an array and the system supports up to 60 axes. Below a sample is shown about a setting with different devices.

AxAddr	Device Code	Internal Ax	BusOffset	Node-Id	Explanation
1	0	1	0	0	Internal Axis to show LINR/LINA command
2	0	1	0	0	Internal Axis to show LINR/LINA command
3	3	0	1000000	1	Technosoft EtherCAT (fist in the EtherCAT Bus)
4	2	0	0	5	Technosoft CAN
5	1	0	0	1	Nanotec CAN
6	1	0	0	2	Nanotec CAN
7					
…					
60					

3.1 AxAddr

The first internal parameter is the AxAddr this number is automatically filled in at start-up and represents the axis address number. This axis address is used in the command string to select the axis. Further parameter that had to set when setting up the system are

- **Internal Ax**
- **BusOffset**
- **Node-Id**

3.2 Internal Axes

The parameter “internal Ax” is a bit to set this axis as an internal axis. Please make sure that all marked internal axes are at the beginning because the AxAddr is used for the internal axis number.

The idea behind the internal axes is that those axes can be controlled from the MasterMACS directly. The CAN or EtherCAT drives are then used in CSP (cyclic synchronous position) or CSV (cyclic synchronous velocity) mode. This allows that the MasterMACS can execute the profile of the movement. Doing so the

features as of a MACS controller can be used. The features are CAM curves, path moves, kinematics, jerk limited moves, or simply synchronous moves among axes.
To show how to use internal axes a command for a combined LINA and LINR move is implemented. Check for Command 90.

3.3 BusOffset

BusOffset defines the used bus for the device. CAN1, CAN2 or EtherCAT.
Because the MasterMACS offers two CAN busses there is a BusOffset value of 0 when the device is connected on the first CAN bus. The value must be 100'000 when the device is on the second CAN bus. Is the device connected to the EtherCAT bus a BusOffset of 1'000'000 is used.

3.4 Node-Id

Node-Id is the CAN-Id of the device. On the EtherCAT bus it is the position along the bus.

3.5 Device Codes

The device code is an internal parameter that holds the information about the device. This is important because not all device behave the same way. The following devices are defined:

```
#define C_DC_NOT_DEFINED          0
#define C_DC_NANOTEC              1
#define C_DC_TECHNOSOFT_CAN       2
#define C_DC_TECHNOSOFT_EC        3
#define C_DC_COBLEY_CAN           4
```

3.6 Position UserUnit Factor

The "Position UserUnit factor" is an internal parameter that holds the information on how the "Target Position" and "Offset" are scaled and stored in SDO objects.The reason for this is that we can only store integer values in the DIM arrays.

1 = 1 (the positions are stored in mm)
10 = 10 (the positions are stored in 10^{th} of mm)
100 = 100 (the positions are stored in 100^{th} of mm)
1000 = 1000 (the positions are stored in 1000^{th} of mm)

4 Commands and Parameters

TCP Cmd	DeviceCode	Internal parameter	Internal only	SDO (ACK/NAK)	Read/ Write	Unit	Explanation
00	**Move**	-		0x6040	W	1 = Absolute move 2 = Relative move 3 = Velocity move 8 = Stop move 9 = Home move	This command is a whole sequence
02	**Target Position**	6*		(0x607A)	RW	UserUnits	Target position is converted before writing SDO
04	**Inhibit**	-		0x6040	RW	0 = Power OFF 1 = Power ON	This command is a whole sequence. Enable: If PowerCycle then NodeReset and WriteParameter before enable sequence
05	**Velocity**	7*		(0x6081)	RW	UserUnits / sec	
06	**Acceleration**	8*		0x6083 (0x6084)	RW	UserUnits / sec^2	
07	**Mode of operation**	-		0x6060	RW	1 = profile position 3 = profile velocity 6 = Homing mode	This command is a whole sequence
09	**Quick-Stop option**	-		0x605A	RW	-	
10	**Status**	-		0x6041	R	-	
11	**Detailed Error**	-			R	-	
12	**Position**	-		0x6064	R	UserUnits with offset and inversion	Nanotec reads 0x6062
13	**In Position Window**	-		0x6067	R	UserUnits	In Position Window
14	**Actual Velocity**	-		0x606C	R		
15	**Mode of operation display**	-		0x6061	R	-	
16	**Reset Node**	-		-	W	-	Reset Node with NMT (CAN only) Restart CAN
17	**Reset Error**	-		0x6040	W	-	Error reset in cmd-word
18	**General Error**	-			R	-	
19	**Device Code**	2		-	RW	-	
20	**Scale_N**	10		-	RW	-	

21	**Scale_Z**	11		-	RW	-	
22	**Axis Inversion**	12		-	RW	1=normal / -1=inverted	
23	**Limit Positive**	13*		0x607D /2	RW	UserUnits	
24	**Limit Negative**	14*		0x607D /1	RW	UserUnits	
25	**Offset**	15*		-	RW	UserUnits	Offset position is converted before writing SDO
26	**Max Velocity**	16*		0x607F	RW	UserUnits / sec	Only Internal for Technosoft and Copley
27	**Max Acceleration**	17*		0x60C5	RW	UserUnits / sec^2	
28	**QuickStop Accel**	18*		0x6085	RW	UserUnits / sec^2	
29	**Slave Axis (AxAddr)**	19		-	RW	-	
30	**Slave Axis direction**	20		-	RW	1=normal / -1=inverted	
31	**Possible Modes of operation**	21		-	R	1 = profile position 2 = profile velocity 3 = both	
32	**Dynamic Limits**	22		-	R	0 = static / 1=dynamic	
33	**Display Limit Positive**	-		0x607D/2 (0x607D/1)	R	UserUnits	Calculated with offset and axis inversion
34	**Display Limit Negative**	-		0x607D/1 (0x607D/2)	R	UserUnits	Calculated with offset and axis inversion
40	**Homig Mode**	25		0x6098	RW		
41	**Homing Velocity (switch)**	26*		0x6099 / 1	RW	UserUnits / sec	
42	**Homing Velocity (index)**	27*		0x6099 / 2	RW	UserUnits / sec	
43	**Homing Acceleration**	28*		0x609A	RW	UserUnits / sec^2	
50	**Read Inputs**	-		0x60FD	R	-	
52	**Set Digital Outputs**	-		0x60FE / 1	RW	-	
53	**Output function**	30		0x60FE / 2	RW	-	
54	**Output delay time**	31		-	RW	ms	
56	**Control Brake**	32		-	RW	1=control brake 0=no controlled brake	Different handling of “enable” and “switched on”
60	**Encoder Type**	35		-	RW	0=INC, 1=SSI, 2=BiSS	
61	**Encoder Position**			0x6064	R	System units	
62	**Following Error setting**	36		0x6065	RW	-	

63	**Encoder direction**	37		0x200D	RW	-	Only Nanotec
80	**Set SDO Index**	38		-	RW	0xXXXXSSLL	Set index/subindex/len for SDO access
81	**Perform SDO Access**	-		0xXXXX / XX	RW	-	perform SDO access to given SDO object. (see Cmd 80)
82	**Motion /Target reached**	-		0x6041	R	0 = motion / 1 = in target	Direct status-bit access
83	**Endswitch hit**	-		0x6041	R	0 = no hit / 1 = switch hit	Direct status-bit access
84	**Error Bit**	-		0x6041	R	0 = Ok / 1= error	Direct status-bit access
85	**Write Parameters**	-		-	W		Write all parameters to controller
86	**Moving Acknowledgement**	-		-	RW	0 = Not started 1 = started	Set 0 before moving
90	**Combined Move** 0 = stop combined move 1 = LINR move 2 = LINA to zero	-		-	W	-	For demonstration with internal axes
91	**Debug Print**	-		-	RW	1=active / 0=inactive	Global parameter; default= 0
99	**SW Version**	-		-	R	-	PSI command interpreter Version …

*internally stored as an integer value, the factor is given in parameter “Position UserUnit factor”.

a. Interpreting Statusbits (TCP Command R10)

Bit Number	Description
15	Manufacturer specific (not to use)
14	Manufacturer specific (not to use)
13	Manufacturer specific (not to use)
12	Manufacturer specific (not to use)
11	Internal Limit Active (current, voltage, velocity or position)
10	Target reached (This bit is set when the drive is finished running a trajectory. The bit is not cleared until a new trajectory is started.)
9	Remote (Is 1 when drive is controlled by the CANopen interface.)
8	Manufacturer specific (not to use)
7	Warning (1 when a warning occurs)
6	Switch On Disabled
5	Quick Stop. When this bit is zero, the drive is performing a quick stop.

4	Voltage Enabled
3	Fault. If set, a fault condition is or was present in the drive.
2	Operation Enabled. Set when the drive is enabled.
1	Switched On
0	Ready to switch on

b. Interpreting Detailed Error Register (TCP Command R11)

Bit Number	Description
15	STO fault (STO input is on disable state)
14	
13	Under-Voltage
12	Over-Voltage
11	Over temperature drive
10	Over-current
9	Negative Software Limit
8	Positive Software Limit
7	Negative Limit Switch
6	Positive Limit Switch
5	Feedback Error
4	Communication Error
3	Following Error
2	Encoder Error (Collective error BISS / Encoder broken wire
1	Short Circuit
0	

c. Interpreting General Error Register (TCP Command R18)

Bit Number	Description
7	Manufacturer specific (do not use)
6	Reserved (always 0)
5	Manufacturer specific (do not use)
4	Communication Error
3	Temperature Error
2	Voltage Error
1	Current Error

0	Generic Error

2 Error Handling

Any error will be clear automatically and a transition in the RUN state is performed.

a. Common Aposs Error Numbers

Number	Name	Description	Group
8	ERROR_TRACK	Track error	Aposs-Error
57	ERROR_STATE_MACHINE	State machine error	Aposs-Error
76	ERROR_ARRAY_SIZE	Array size is not correct.	Aposs-Error
89	ERROR_CAN_SDO	SDO send or receive error	Aposs-Error
91	ERROR_CURVE	Curve Array is not correct.	Aposs-Error
1-103	ERROR_APOSS	Unusual Aposs Error (please see Aposs help file)	Aposs-Error

3 Further information

a. Enable / Disable EtherCAT

There is a define in the application to enable or disable EtherCAT bus scan. This is important because the MasterMACS generates an error in case of no EtherCAT slaves.

```
#define EtherCAT_Active    0      // EtherCAT Master active = 1; disabled = 0;
```

b. Integrating CANopen IO Module

For more digital inputs and outputs a CANopen IO module/device can be integrated in the CAN bus. The CANopen device must have the same CAN baudrate settings as the other devices within the CAN bus.

In the TCP-IP to CANopen application a MACS internal Busmodule is used to do all the communication to and from the CANopen IO device in the background. Furthermore, the mapping of the communication object to the virtual inputs or virtual outputs is done. This gives the user the ability to use standard AposSC commands for accessing those inputs and outputs the same way as internal hardware IOs.

The commands are “DigOutputByte()”, “DigOutput ()”, “DigInputByte()”, “DigInput()”. The DigOutputByte coming from the CAN bus starts at byte 4. The DigOutput number starting with number 33.
The DigInput number starts with 33 and the DigInputByte with byte 4.

In the application there is a function “void SetupCAN_IOmodule(long module_no)” that can be called with “SetupCAN_IOmodule(0);” Call this function with the parameter of an available module number. The module number must be unique within the whole application program. The Node-Id of the CANopen IO module is given in the #define CAN_NODE_ID_IO_MODULE. An offset of 100000 is used when the CAN device is connected on CAN2.

c. MiniMACS as CANopen IO Module

When using a MiniMACS (001586) as a CANopen IO Module, a special application must be loaded to the MiniMACS. This application “IO_Module_SIM.mc” simulates all the CiA401 functionality. If this application is booted the “device type” (0x1000) is also changed. The device type of an IO module is different from a MACS so AposslDE does not connect to this MiniMACS anymore. For connecting with AposslDE use the MiniMACS DIP switch 5,6,7 in (ON) position CAN-Id 0x7n and reboot. A node-Id in this range is not possible when using as CANopen IO module.