POSITIONING AND CONTOURING CONTROL SYSTEM

APCI-8008 – ETHERCAT

(APCI-8008 / APCIe-8008 / APCIe-8008-EC)

Last updated: 21/09/2021, from disk V2.53VO
Board revision: APCI-8008 Rev. C
APCIe-8008 Rev. A
APCIe-8008-EC Rev. 0
Rev. 05/052022

www.addi-data.com

1 EtherCAT master

The APCI-8008-EC provides the user with the familiar function of the APCI-8008 with the option of connecting EtherCAT slave components. These can be input/output modules or drive elements.
The family of the APCIx-8008 board includes three products which allow for EtherCAT connection:

APCI-8008 and BOB-3100
APCIe-8008 and BOB-3100
APCIe-8008-EC.

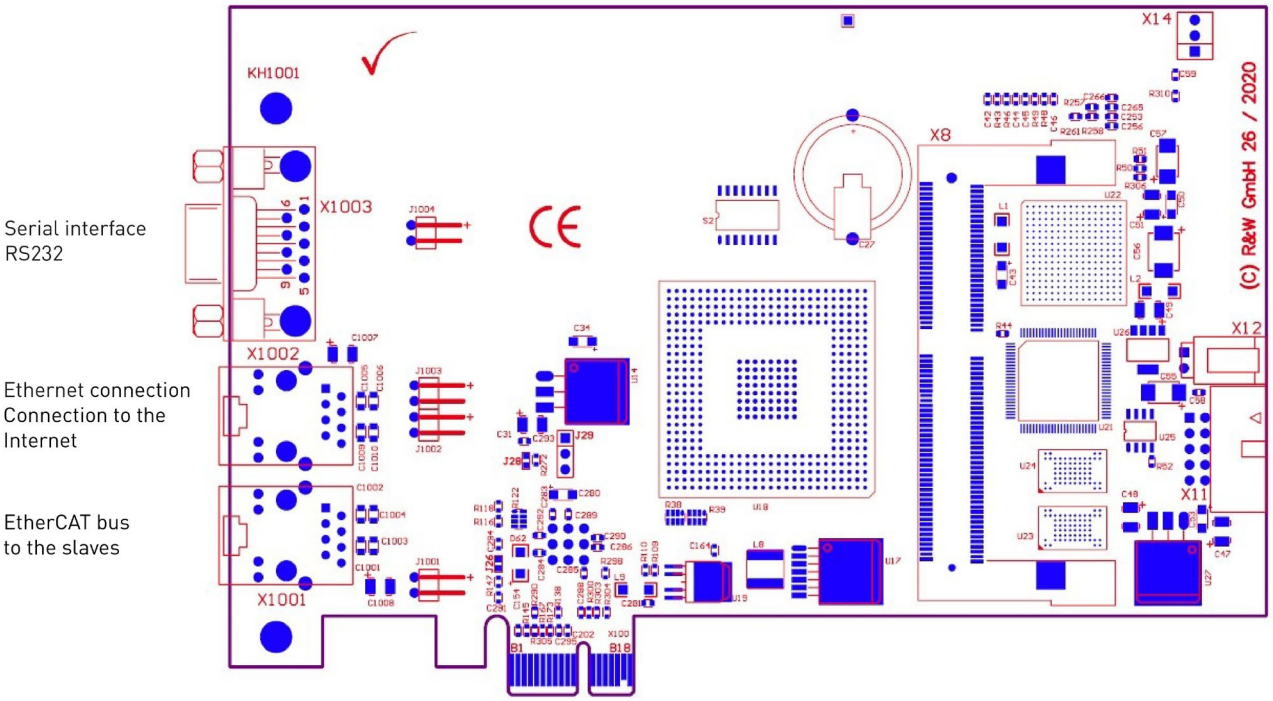
1.1 Brief description

The components (slaves) to be used in the EtherCAT bus must be implemented in the existing system software. For this reason, the suitability of the intended EtherCAT components must be checked before each use and, if necessary, be integrated by the manufacturer of the APCIx-8008 into the system software.
The boards APCI-8008 and APCIe-8008 also support a mixed version of EtherCAT components and conventional drive components. This means that the standard interfaces (Analog Out, Incremental encoder, I/O interface) can be operated along with EtherCAT components on the same system.
The cycle time of the control system is fixed to 1 ms and cannot be changed.

2 Hardware requirements

2.1 APCIe-8008-EC

Figure 2-1: APCIe-8008-EC connections



2.2 APCI-8008 and APCIe-8008

In order to use an APCI-8008 / APCIe-8008 for EtherCAT applications, the components for the Ethernet connection must be fitted. These include, for example, connectors X7 and X9, which represent the Ethernet interfaces. For the connection of the APCI-8008 to the EtherCAT bus, the break-out board BOB-3100 must be used. The following compounds are to be prepared:

APCI-8008 X7 (ETH0)	←→	BOB-3100 X11	X1 Ethernet connection for possible Internet connection
APCI-8008 X9 (ETH1)	←→	BOB-3100 X12	X2 EtherCAT connection
APCI-8008 X10 (RS232)	←→	BOB-3100 X13	X3 serial interface BOB-3100

Notice: The Ethernet connections to X7 and X9 of the APCI-8008 must be made with the supplied ribbon cables, since standard cables do not guarantee the necessary characteristic impedance. Any jumpers in connectors X7 and X9 must be removed before using the connector.

2.3 Location of the connectors on BOB-3100 and APCI-8008

Figure 2-2: BOB-3100 connections

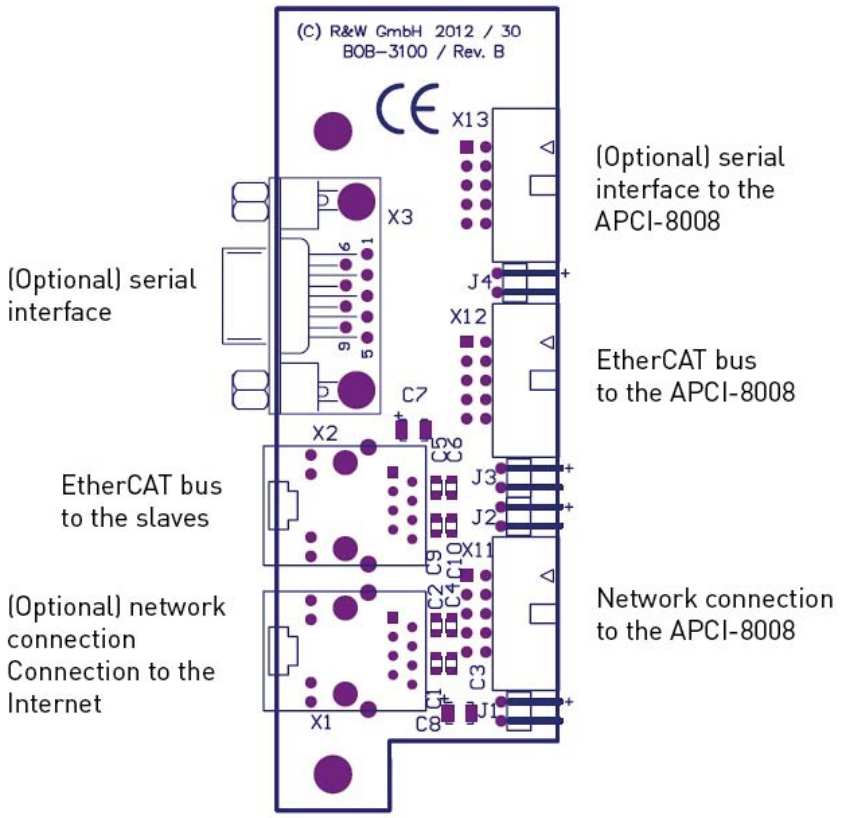
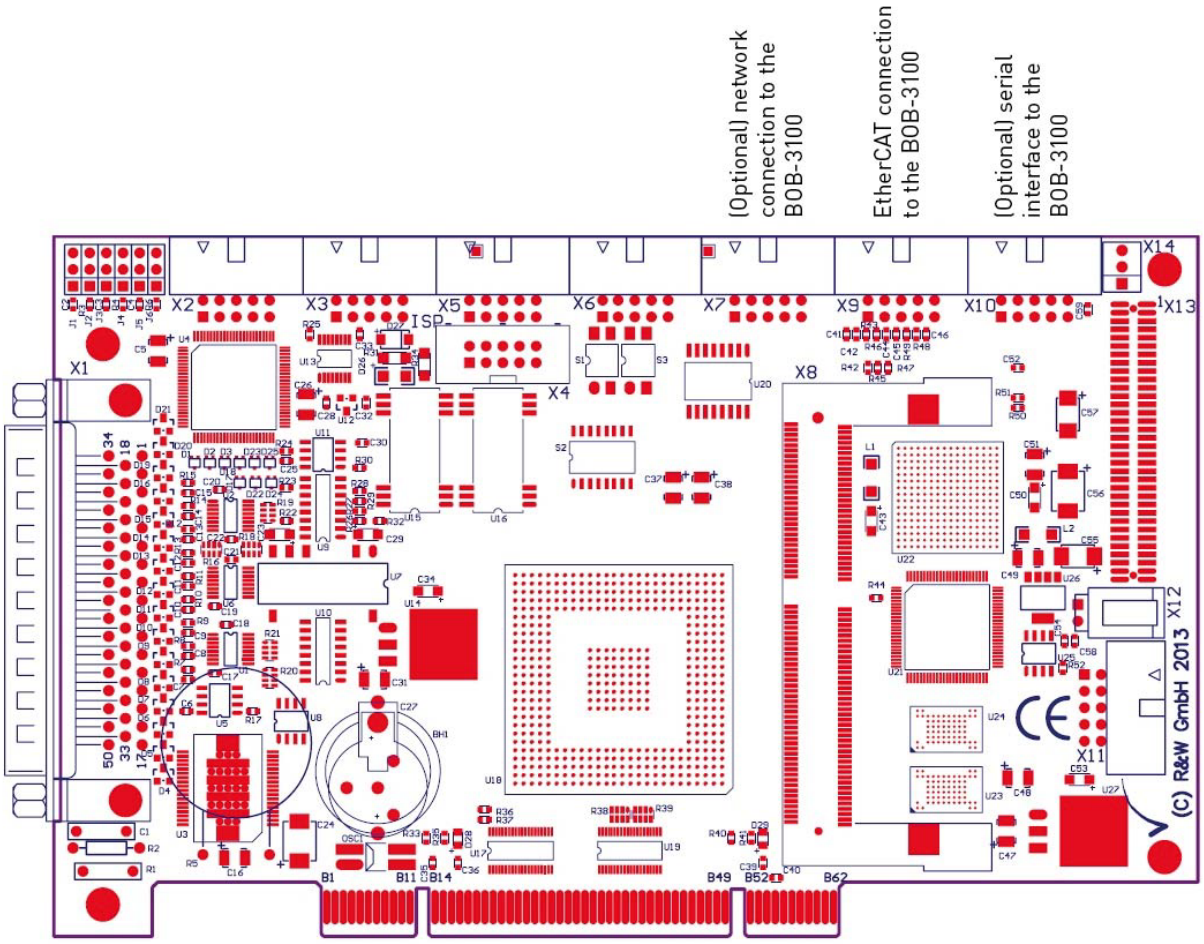


Figure 2-3: APCI-8008 connections



3 Software requirements

The APCI-8008-EC must be configured so that Linux and the Linux version of the APCI-8008 RWMOS.ELF run on the board. This version boots automatically after switching on the PC or after a system reset in fwsetup.exe.
The call of the DLL function BootFile is not allowed here and leads to a system failure.

As with the standard version of the APCI-8008, different RWMOS variants also exist for the EC version. Here, an update via an FTP server is possible. To do this, X7 of the APCI-8008-EC must be connected to X11 of the BOB-3100. The network connection for Internet access is made at connector X1 of the BOB-3100 (RJ-45).

Notice: When booting the APCI-8008-EC Linux kernel, the network connection must already exist; otherwise, no Internet update is possible in the respective session.

4 Commissioning of the EtherCAT system

4.1 fwsetup.exe program

4.1.1 fwsetup monitor screen – HVC0 tab

In contrast to the so-called monitor screen in the standard version of the APCI-8008, the HVC0 tab is available here in fwsetup. This represents a Linux console, with which the Linux kernel of the APCI-8008-EC can be accessed. Alternatively, the board can also be accessed via Telnet using a terminal program.
The required IP address can be determined on the HVC0 tab via ifconfig. Per user root, the login can be done with the default password “mcu”.
With a terminal program, you have the option of a larger display because fwsetup truncates characters outside the display area. Nevertheless, access via fwsetup is sufficient for many applications.

Notice: You can make the truncated areas visible by transferring the monitor content to the clipboard (with the mouse button: "Copy to ClipBoard") and pasting the clipboard into an editor.

4.1.1.1 Displaying RWMOS version and options

The diagnosis of the used RWMOS variant can be done with the following command:

/opt/mcu/info.sh

This returns e.g. the following output:

Fri Sep 14 10:05:05 CEST 2018
This is rwmoslkm with build number 2.5.3.144-5e77b1c, created at Sep 14 2018 09:26:29.
Compiled in options are: APCI-8008-LINUX [3C5, 18A, EV, SSI, RESOURCE, SCANNER, ELCAM, GEAR, SPLINE, TC, LANGHAM, EXTENDED_DIAG, PIDFIDLE, MULTICONTROL, EC].
Up to 5 axis are supported, cycle time is: 1000 micro seconds.

Here, the characteristic and prerequisite for the EtherCAT option is the operating system option "EC".

4.1.1.2 Further fwsetup diagnostic options

At this point, there are many input options for system configuration and diagnosis, such as the EtherCAT status display:

ethercat diag

A grouped description of these options can be found in Chapter 6.

4.1.1.2.1 Assigning an alias address

Each slave can be assigned a unique alias address. This value is then stored as resident on the respective slave and can be retrieved / verified when the application is started. With it, the cabling of the bus structure can be checked, for example.

Example:

ethercat alias –p0 101

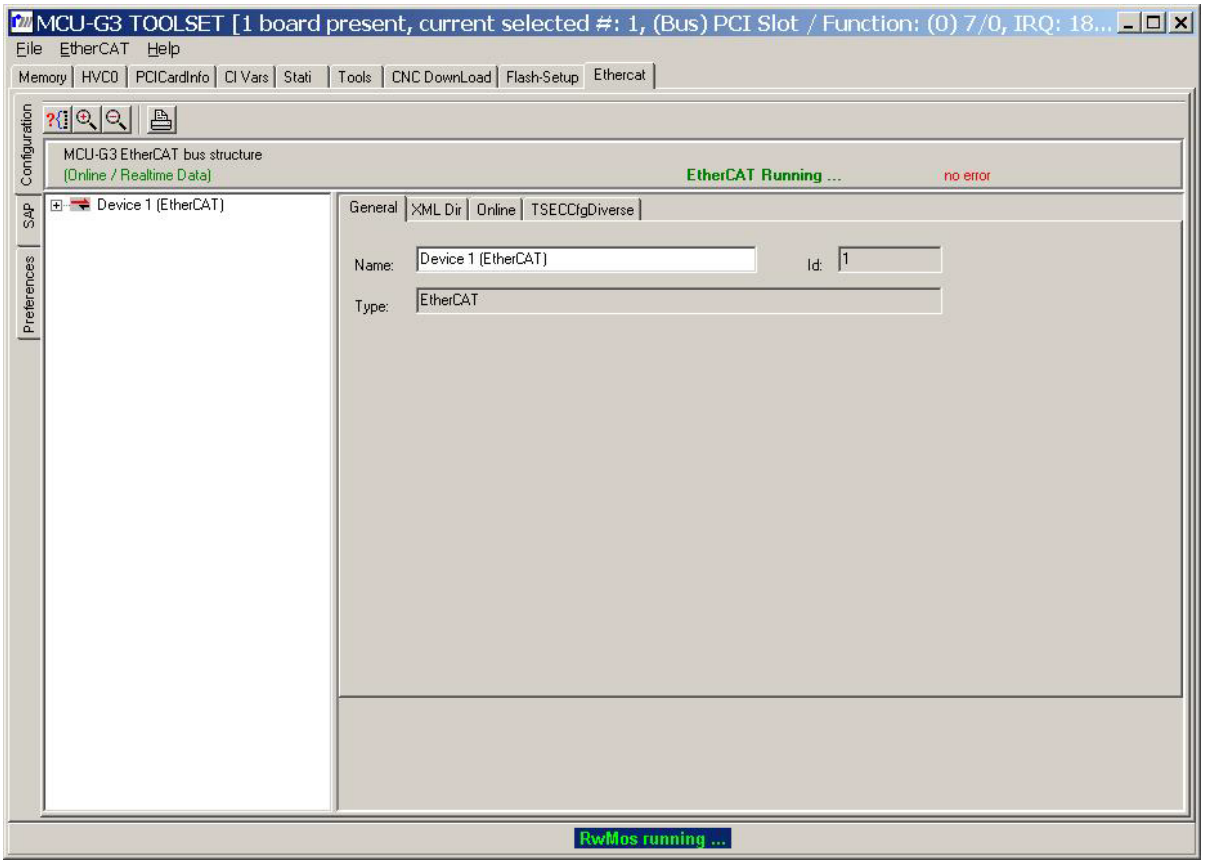
-p0 means that the alias address is assigned to the first slave in the bus.
In this example, 101 is the alias address. It is displayed in the second column via ethercat diag.
Reading the alias address during runtime is addressed with the resource Dev.# 700 hex Subindex 1 using the universal object interface. The consecutively numbered slave position in the bus starting from 0 must be indicated in the index.

4.1.2 fwsetup EtherCAT tab

With the aid of the TOOLSET program *fwsetup.exe*, the EtherCAT bus structure connected to the APCI-8008 can be automatically detected, as long as the connected bus is in a proper state. The additional configuration options for the bus can be made on the [Ethercat] tab. There are currently three different pages on which important information about the current bus structure and status can be determined. Various data contained herein are required for SAP or PCAP programming.

Click on the button  to read the bus structure. This can take some time. By clicking on the plus sign of the Device 1 node, the bus structure can be expanded. Device 1 represents the EtherCAT master of the APCI-8008.

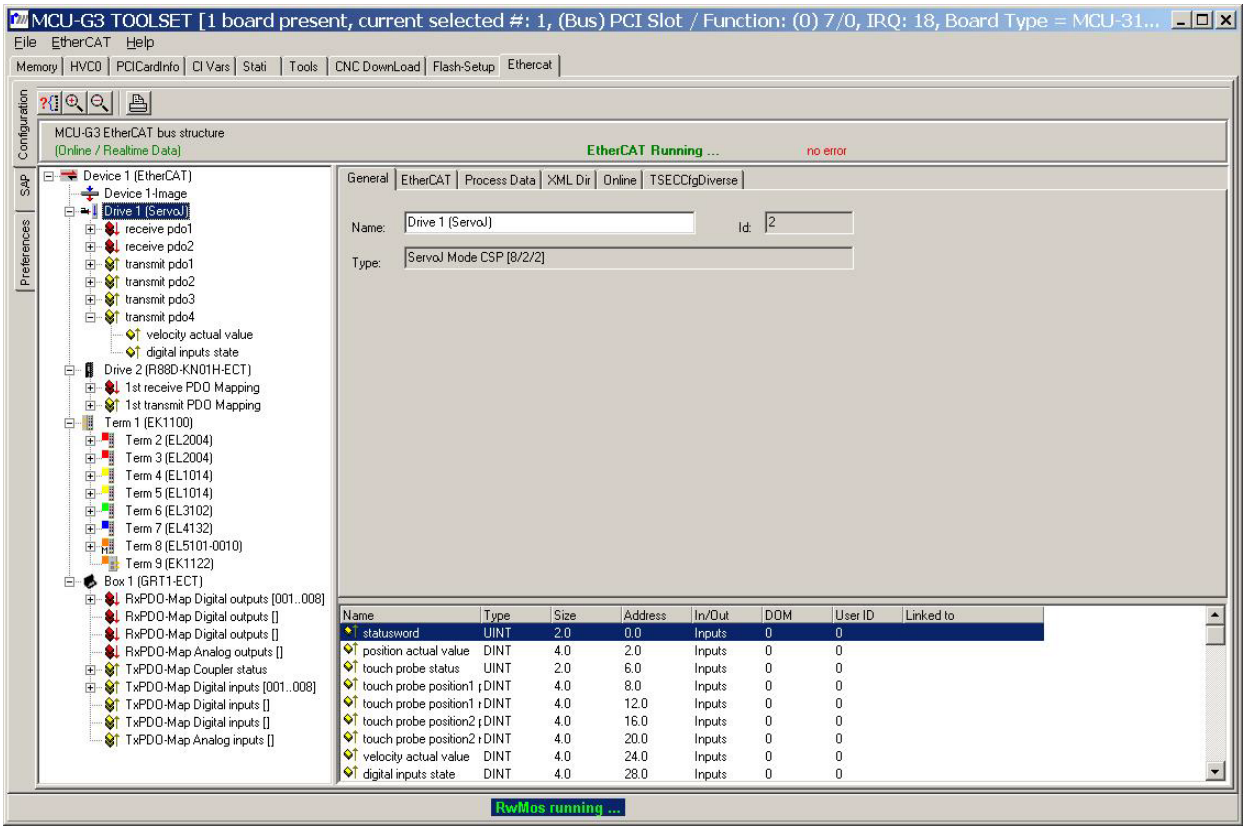
Figure 4-1: EtherCAT Configuration



4.1.2.1 EtherCAT page: Configuration

On the page "Configuration", the bus structure connected to the BOB-3100 can be graphically displayed in tree form (see the following screenshot). The bus couplers (bus nodes) with device identifier and plain text description are listed in the top hierarchy level. Underneath, the individual modules are displayed with their specific properties such as device ID, description, access type and word size.

Figure 4-2: EtherCAT Configuration: Tree form



To activate this screen page, the operating system rwmos.elf must already be started on the board and the bus must be connected (Online Mode, RwMos running).
The following table explains the graphical operating icons (buttons) of the screenshot above.

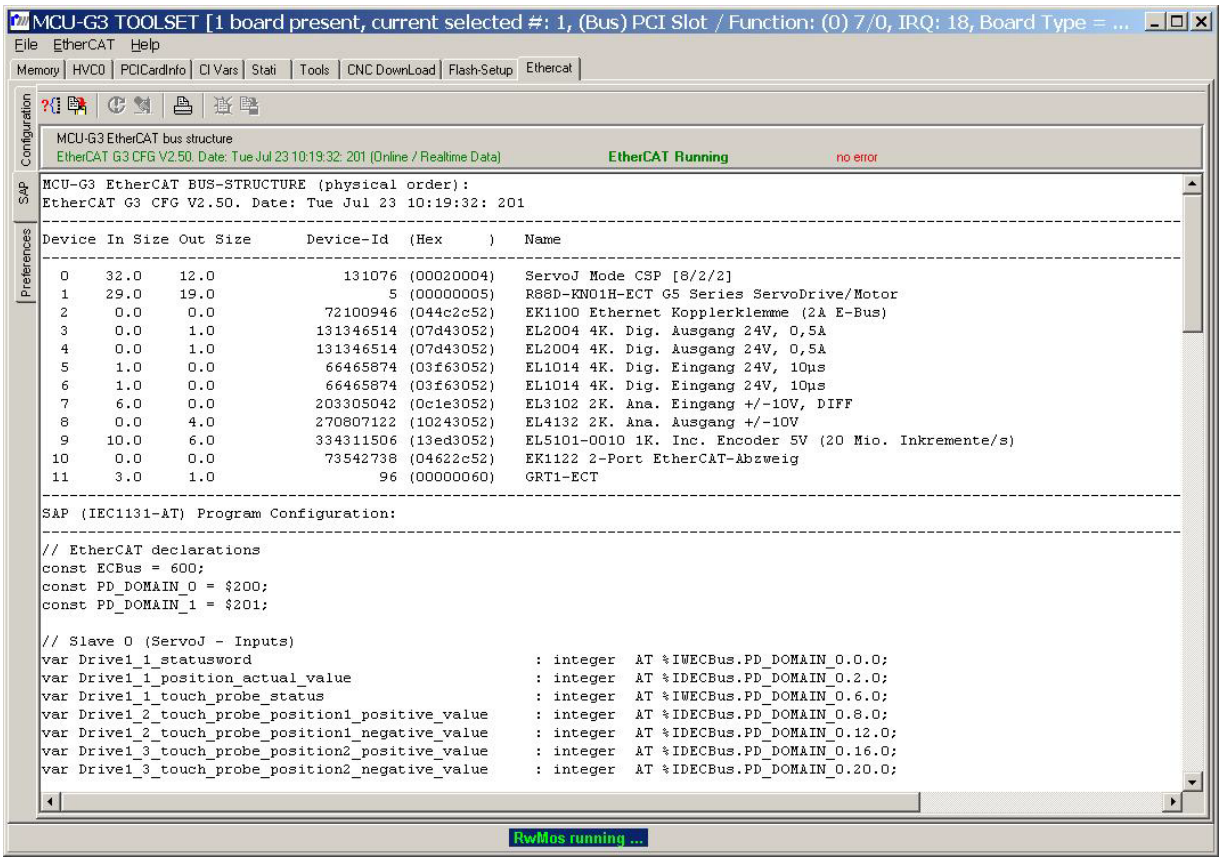
Table 4-1: Meaning of the icons on the [EtherCAT][Configuration] page

Icon	Meaning
	Read the last determined EtherCAT configuration of the APCI-8008. The data collected during the last EtherCAT initialisation are evaluated. **Notice:** This button may only be pressed when the initialisation of the EC bus is finished. This is generally the case when all subscribers are in operational mode. Depending on the number of subscribers, this can take up to a minute or more. Failure to do so may cause error messages to appear or variable names to be displayed incorrectly (for example, "unknown"). In this case, fwsetup.exe should be stopped and restarted.
	The current device directory tree is expanded. All available information is displayed.
	The current device directory tree is collapsed. Only bus nodes are displayed.
	The current configuration can be output to a printer for documentation purposes.

4.1.2.2 EtherCAT page: SAP

On the SAP page, the bus structure connected to the BOB-3100 can be displayed in text form by clicking on  (see the following screenshot).

Figure 4-3: EtherCAT SAP



In order to activate this screen page, the operating system rwmos.elf must already be started on the board (Online Mode, RwMos running).

4.1.2.2.1 Explanation of the graphical operating icons

The following table explains the graphical icons (speed buttons) of the screenshot shown in Figure 4-3.

Table 4-2: Meaning of the icons on the [EtherCAT][SAP] page

Icon	Meaning
	see Table 4-1

	With the help of this button, the AT specifiers introduced according to IEC1131 can be copied to the Windows clipboard for further use in an SAP program. After pressing this button, the Windows clipboard for the above bus structure contains e.g. the following information:
	
	// EtherCAT declarations
	const ECBus = 600;
	const PD_DOMAIN_0 = $200;
	const PD_DOMAIN_1 = $201;
	
	// Slave 0 (ServoJ - Inputs)
	var Drive1_1_MPRO_402_Status : integer AT %IWECBus.PD_DOMAIN_9.10.0;
	var Drive1_1_MPRO_402_ActualTo : integer AT %IWECBus.PD_DOMAIN_9.12.0;
	var Drive1_1_MPRO_402_PositionActua : integer AT %IDECBus.PD_DOMAIN_9.14.0;
	var Drive1_2_MPRO_INPUT_S : integer AT %IDECBus.PD_DOMAIN_9.18.0;
	// Slave 1 (R88D-KN01H-ECT - Inputs)
	var Drive2_1_Error_code : integer AT %IWECBus.PD_DOMAIN_9.34.0;
	var Drive2_1_Statusword : integer AT %IWECBus.PD_DOMAIN_9.36.0;
	var Drive2_1_Position_actual_value : integer AT %IDECBus.PD_DOMAIN_9.38.0;
	var Drive2_1_Torque_actual_value : integer AT %IWECBus.PD_DOMAIN_9.42.0;
	...
	
	These definitions can be used for variable declaration in an SAP program.

	The current configuration can be output to a printer for documentation purposes.

4.1.2.2.2 Explanation of the screen text output EtherCAT SAP

Figure 4-4: Screen text output EtherCAT SAP

```
MCU-G3 EtherCAT BUS-STRUCTURE (physical order):
EtherCAT G3 CFG V2.50. Date: Tue Jul 23 10:19:32: 201
-----------------------------------------------------------------------------------------------------------------------------
Device In Size Out Size      Device-Id  (Hex     )   Name
-----------------------------------------------------------------------------------------------------------------------------
  0    32.0    12.0              131076 (00020004)   ServoJ Mode CSP [8/2/2]
  1    29.0    19.0                   5 (00000005)   R88D-KN01H-ECT G5 Series ServoDrive/Motor
  2     0.0     0.0            72100946 (044c2c52)   EK1100 Ethernet Kopplerklemme (2A E-Bus)
  3     0.0     1.0           131346514 (07d43052)   EL2004 4K. Dig. Ausgang 24V, 0,5A
  4     0.0     1.0           131346514 (07d43052)   EL2004 4K. Dig. Ausgang 24V, 0,5A
  5     1.0     0.0            66465874 (03f63052)   EL1014 4K. Dig. Eingang 24V, 10µs
  6     1.0     0.0            66465874 (03f63052)   EL1014 4K. Dig. Eingang 24V, 10µs
  7     6.0     0.0           203305042 (0c1e3052)   EL3102 2K. Ana. Eingang +/-10V, DIFF
  8     0.0     4.0           270807122 (10243052)   EL4132 2K. Ana. Ausgang +/-10V
  9    10.0     6.0           334311506 (13ed3052)   EL5101-0010 1K. Inc. Encoder 5V (20 Mio. Inkremente/s)
 10     0.0     0.0            73542738 (04622c52)   EK1122 2-Port EtherCAT-Abzweig
 11     3.0     1.0                  96 (00000060)   GRT1-ECT
-----------------------------------------------------------------------------------------------------------------------------
SAP (IEC1131-AT) Program Configuration:
-----------------------------------------------------------------------------------------------------------------------------
// EtherCAT declarations
const ECBus = 600;
const PD_DOMAIN_0 = $200;
const PD_DOMAIN_1 = $201;

// Slave 0 (ServoJ - Inputs)
var Drive1_1_statusword                          : integer  AT %IWECBus.PD_DOMAIN_0.0.0;
var Drive1_1_position_actual_value               : integer  AT %IDECBus.PD_DOMAIN_0.2.0;
var Drive1_1_touch_probe_status                  : integer  AT %IWECBus.PD_DOMAIN_0.6.0;
var Drive1_2_touch_probe_position1_positive_value : integer  AT %IDECBus.PD_DOMAIN_0.8.0;
var Drive1_2_touch_probe_position1_negative_value : integer  AT %IDECBus.PD_DOMAIN_0.12.0;
var Drive1_3_touch_probe_position2_positive_value : integer  AT %IDECBus.PD_DOMAIN_0.16.0;
var Drive1_3_touch_probe_position2_negative_value : integer  AT %IDECBus.PD_DOMAIN_0.20.0;
var Drive1_4_velocity_actual_value               : integer  AT %IDECBus.PD_DOMAIN_0.24.0;
var Drive1_4_digital_inputs_state                : integer  AT %IDECBus.PD_DOMAIN_0.28.0;
// Slave 1 (R88D-KN01H-ECT - Inputs)
var Drive2_1_Error_code                          : integer  AT %IWECBus.PD_DOMAIN_0.32.0;
var Drive2_1_Statusword                          : integer  AT %IWECBus.PD_DOMAIN_0.34.0;
var Drive2_1_Position_actual_value               : integer  AT %IDECBus.PD_DOMAIN_0.36.0;
var Drive2_1_Torque_actual_value                 : integer  AT %IWECBus.PD_DOMAIN_0.40.0;
var Drive2_1_Modes_of_operation_display          : integer  AT %IBECBus.PD_DOMAIN_0.42.0;
var Drive2_1_Touch_probe_status                  : integer  AT %IWECBus.PD_DOMAIN_0.43.0;
var Drive2_1_Touch_probe_pos1_pos_value          : integer  AT %IDECBus.PD_DOMAIN_0.45.0;
var Drive2_1_Touch_probe_pos2_pos_value          : integer  AT %IDECBus.PD_DOMAIN_0.49.0;
var Drive2_1_Digital_inputs                      : integer  AT %IDECBus.PD_DOMAIN_0.53.0;
var Drive2_1_Velocity_actual_value               : integer  AT %IDECBus.PD_DOMAIN_0.57.0;
// Slave 5 (EL1014 - Inputs)
```

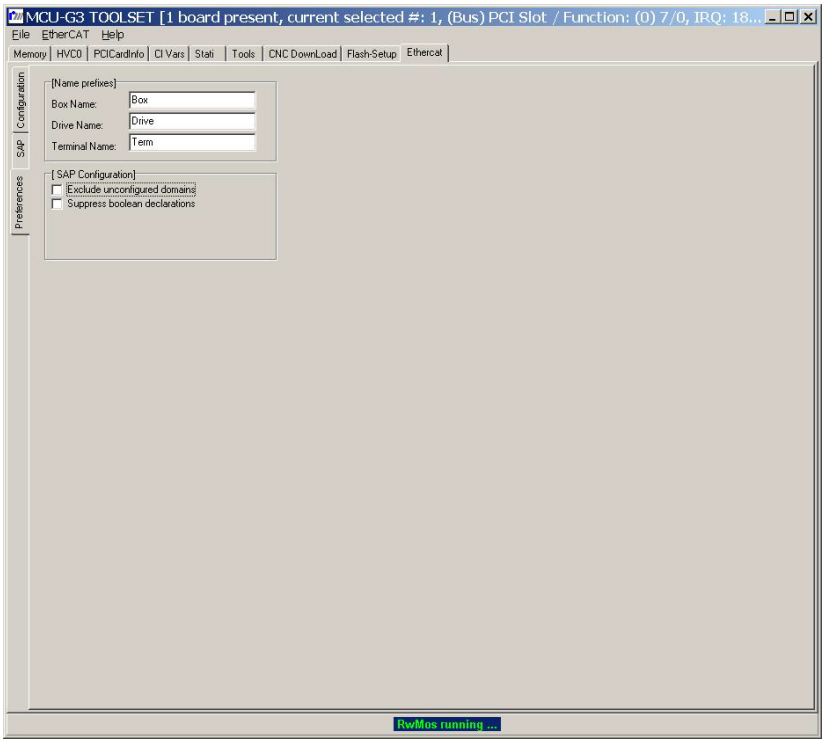
All existing EtherCAT devices are given a consecutive number (*Device* column). This number corresponds to the actual position in the bus structure. The column *Device-Id* describes the device function. Normally, the ID codes are unique identifiers of the EtherCAT modules in decimal and hexadecimal notation. The *Name* column describes the respective module in plain text.
For SAP and PCAP programming, useful information is also contained in this screen text output. First of all, this includes all type definitions that can be used to address the respective EtherCAT devices. The coding of the AT-*Specifiers* are the declarations of the system variables for the SAP programming. If required, these provide the necessary information for the object access by means of the PCAP programming. In the SAP programming language, access to the I/O level is controlled using the keyword AT, based on the IEC1131 programming language (see manual “Universal Object Interface”, Chapter 2.2).
The *access* type (first letter after AT % (I or Q)) can be used to determine how the respective system variable may be accessed, i.e. read (I) or write (Q) access.

Notice: For the above bus structure, there is the SAP demo program "SAP_DEMO_EtherCAT.SRC", in which the flexible and simple EtherCAT communication is demonstrated. This program can be found on the Toolset CD from V2.53VN.

4.1.2.3 EtherCAT page: Preferences

On the page "Preferences", you can define requirements for the names of the SAP variables.

Figure 4-5: EtherCAT Diagnostics



4.1.3 EtherCAT SAP programming

In addition to the PCAP programming method, the additional software functions of the EC option can also be used very effectively by the SAP programming method.

Notice: The EC option should be accessed only from one task. Otherwise, it must be ensured that the EC option is only accessed sequentially (not simultaneously) from several tasks. If this is not guaranteed, this can lead to inexplicable runtime behaviour!

4.1.3.1 EtherCAT access to slave modules via universal object interface

The EtherCAT function can be used by reading and writing the resources detected by fwsetup. With the aid of these variables, it is possible to communicate directly with the individual slaves. The definition of the individual variables depends on the bus structure and is only valid as long as the bus has not been changed. The variable access is controlled with the help of the AT mechanism, based on the IEC1131 programming language. This is described in detail in the manual “Universal Object Interface” (Chapter 2.2). The AT specifiers generated automatically here correspond to the process data variables (PDOs) of the respective slave.

Notice: In the SAP user program, the EtherCAT bus should be monitored, e.g. through status queries in an event handler routine.

4.1.3.2 Access to service data objects (SDOs)

The universal object interface allows access to service data objects of the individual slaves. However, the individual AT specifiers must be declared manually with the help of the respective product documentation. For this, the following guidelines apply.

For the bus number, EcBus must be indicated. For the device number, the respective slave index must be entered. A maximum of 512 slaves are supported (0..511). On the one hand, it is of course assumed for the use of these variables that the bus structure is strictly adhered to and not changed. On the other hand, it is also possible to control the expected bus structure by reading out and verifying the identifier of the individual subscribers one after the other.
For Index and SubIndex, the respective SDO index / subindex must be entered. The respective product documentation provides information about this.

When accessing SDOs, it is important to ensure that SDOs are never accessed at the same time from different locations, as the execution of these commands can only be completed in the following cycle. If an ongoing access to an SDO in one task is followed by another access from another task or by PCAP, the process is disrupted with unforeseeable consequences.

Furthermore, it is important to note that service data objects must generally not be written on cyclically. If it is so-called backup data, which is kept resident in the flash memory of the slave when being written on, destruction of the slave module is otherwise possible because the flash area must not be written on as often as desired. This can be remedied e.g. by reading a variable and writing the respective initialisation value only on it if the value is invalid.

Moreover, the following needs to be considered:
After starting the EC bus or after restructuring, read access to all SDOs in the entire EC bus takes place. During this phase, which may take several minutes, read or write access to SDOs from a program is practically not possible, as the respective commands are processed sequentially. Thus, a pre-initialisation of SDOs by means of access via the universal object interface is not really suited. Instead, from RWMOS version 2.5.3.154, there is the possibility to pre-initialise SDOs in the entire bus with the aid of a configuration file.

4.1.3.3 EtherCAT error handling

The event handler EVEC can be used to handle EtherCAT error messages. It responds when a bus error BEF is detected in the interface status register. This error can only be acknowledged, if at all, with a ResetSystem command (rs) at the PCAP level. With this command, however, all SAP tasks are stopped.
The rs command should not be used at the SAP level. After executing the rs command, you have to wait until the bus error flag has been withdrawn. Thereafter, the further program sequence can be initiated.

4.1.4 EtherCAT PCAP programming

The accesses to the individual variables of the EtherCAT area are handled via the universal object interface. To do this, use the list of resources that can be created in fwsetup on the EtherCAT tab on the SAP tab.

4.1.4.1 Special features when accessing service data objects (SDO)

When accessing an SDO from PCAP programming, the return value of the call must always be evaluated. As long as the return value BUSY (2) is supplied, the access is still in progress and must be repeated until success (4) or perhaps an error is displayed.
Again, it is important to note that SDOs may only be accessed from a single location.

4.2 SDO initialisation using configuration data in a file

In order to reliably initialise SDO configuration data, the corresponding data can be indicated in the configuration file “/etc/sdos.conf”.
This data is written on the slaves even before the Operational Mode is reached and also every time after switching on a slave. To use these operations, specific minimum versions of the applied software are required:

rwmos kernel module	V2.5.3.154
mcug3.dll	V2.5.3.131
fwsetup.exe	V2.5.3.39

4.2.1 Structure of the configuration file

The configuration file is a text file that can be created with a common editor.
Data elements are organised in lines and columns. Columns are separated by spaces (blanks or tabs).
The file name and the path are predefined. Upper and lower case must be respected.

4.2.1.1 First line of sdos.conf

The first line contains an SDOSINIT identifier and the version information V1.0.

SDOSINIT V1.0

4.2.1.2 Following lines in sdos.conf

The following lines include information on the SDOs to be initialised. The respective numerical values can be indicated in decimal or hexadecimal form preceded by 0x.

Column #	Content	Note	
1	Slave Vendor Id	Identification of an EtherCAT module	
2	Slave Product Code	Identification of an EtherCAT module	
3	Revision Number	Identification of an EtherCAT module; in case of –1, the revision number is ignored	
4	Slave Index	Index of a module in the EC bus to be initialised (starting with 0); in case of –1, the index is ignored; all identified slave modules in the bus are initialised then	
5	Index	SDO address “Index”	
6	SubIndex	SDO address “SubIndex”	
7	Byte Size	Word size of the SDO in bytes (1, 2 or 4)	
8	Value	Numerical value with which the respective SDO is initialised	

Example:
0x0000066F 0x51505001 –1 –1 0x6085 0 4 100000000

4.2.1.3 Notes

Lines containing notes can be hidden with the “#” sign (without inverted commas).

4.2.2 Saving a configuration file in the APCIx-8008-EC file system

The configuration file is created under the Windows system of the host PC and can be transferred to the motion control board using the program “fwsetup”. The file is then saved as resident in the flash file system of the board. To do so, “fwsetup” must be called when the EtherCAT system is ready for operation. Afterwards, a file can be searched for and installed via the “sdos.conf Installation” button under the “Ethercat/Update” tab, and here again, on the “File Transfer” tab.
Notice: If a motion control board is replaced or the file system of a board is changed through an update function, this file has to be restored on the board. This information must be indicated in the product documentation.

4.3 EtherCAT diagnosis via universal object interface

The EtherCAT technology has proven to be an extremely reliable system. However, data transfers may be disturbed through different reasons. In this chapter, possible ways of diagnosis are shown in order to detect potential errors or handling problems.
With the EtherCAT system of the APCI-8008, it is assumed that all slaves are contained in one sync unit. During the first start, the EtherCAT master scans the EC bus and determines the connected devices. A software diagnosis can be made only after this process is finished.

4.3.1 Working Counters and number of detected slaves in the bus

After scanning the EC bus, the “Expected Working Counters” for domain 0 (input information) and domain 1 (output information) are available. First of all, these can be compared to the values expected from the application and to which they must correspond. If this is not the case, the expected bus configuration is wrong or the bus is significantly disturbed. This may occur, for example, if parts of the machine are switched on too late. In this case, the bus configuration must be corrected and the motion control board be restarted afterwards.
The Expected Working Counters are read with the resource Device # 900 hex, Index 1, SubIndex 0 for domain 0 and SubIndex 1 for domain 1. Both values can be read at once using SubIndex 2, with the Expected Working Counter of domain 0 being stored in the lower 16 bits of the function value and the Expected Working Counter of domain 1 in the higher 16 bits (e.g. 0007 0007 hex).

If the Expected Working Counters correspond to the values required for the application, the current values of the Working Counters can be read with the resource Device # 900 hex, Index 2, Subindex 0, 1 or 2 (see previous paragraph). It may take some time, though, until the current values of the Working Counters correspond to the expected values, because the slaves have to be put in the appropriate state first.
In the event that the bus is interrupted during normal operation due to failure of a slave module or disconnection of the EC bus, for example, the current values of the Working Counters fall back immediately. The Expected Working Counters keep the values determined at boot-up. If there are servo modules with closed position control loops in the remaining active bus, these are instantly opened, as it has to be assumed now that normal operation of the axis system can no longer be ensured.
If in this case of error, the expected bus structure is restored, the current Working Counters reach the expected values again after some time. A continued operation of the whole system without any restart is possible then.
The current values of the Working Counters never outnumber the Expected Working Counters.

By means of the resource Device # 900 hex, Index 3, SubIndex 0, the number of detected slaves in the bus can be read. This piece of information is also an important indicator for a correct existence of the bus structure. Compared to the current Working Counters, this value may be available with a delay. However, this value helps to detect further slaves that are added to the bus during runtime, which is not possible by monitoring the Working Counters.

4.3.2 Lost Frame Counter

The Lost Frame Counter is an error counter of the master module, which can be used as the first indicator for communication problems of the hardware level. The Lost Frame Counter indicates if a data packet sent does not return or returns damaged.
The Lost Frame Counter is read via Device # 900 hex, Index 4, Subindex 0. The error counter is an incremental value and only reset by a restart of the system.

4.3.3 Status of the slaves in the bus

Each slave in the EtherCAT bus has a maximum of 4 EtherCAT ports at which incoming or outgoing Ethernet data traffic is possible. For a slave in the EC bus, normally two ports are reserved; one to the master or previous slave and one to the following slave. With the last slave in the bus or in a bus branch, only one port is active. With a slave in the bus with a bus junction, all 4 ports are active.
For each of these ports, the EtherCAT slave provides an error counter for “Link Lost Counter”, “CRC Error Counter” and “RX Error Counter”. These counters can be read if the slave is accessed via EtherCAT.
By analysing this information, a fault location in the EC bus can be pinpointed or narrowed down. These error counters are accessed via Device # 900 hex according to the following table.

4.4 Reading bus and slave information

The information described below can be accessed using the universal object interface. Many elements are only available for read operations. Access is provided via SAP or PCAP programming.
The bus number for the accesses described here is always ECBus (600 dez.). The indicated data type always needs to be taken into account. Boolean, byte and single words are to be defined as integer in SAP programming and to be operated via wrOptionInt or rdOptionInt in PCAP programming.
This information can be used, for example, to check the connected bus.

4.4.1 Determining general information of the slaves in the bus

In this group of variables, the access variable Index contains the index of the slave device to be accessed (starting from 0). This is also the index that is assigned to the subscribers in the "ethercat diag" command, for example, and that represents the order in the bus.
In SubIndex, a function number is indicated. These functions are described below.

AccessType: Read
DataType: According to the data type indicated in the following table

4.4.1.1 Dev.# 700 hex

This device number can be used to read information about the connected slaves. Write access is not possible here.

SubIndex	Name Data type	Description
0	EXIST integer 32 bit	This variable can be used to determine if a slave exists in the bus. If 0 is returned, the subscriber is not available. The index may take values from 0 to 511. The bus subscribers are listed consecutively starting from 0. The calling function returns an error if a non-existent slave is accessed with subsequent function numbers. The determined parameter is invalid then.
1	ALIAS short int 16 bit	This call returns the assigned alias address of a slave. This value can be used, for example, to verify the bus structure or to check the bus cabling.
3	Vendor Id. integer	This call returns the manufacturer code of the corresponding slave module. This value can be used, for example, to verify the bus structure.

SubIndex	Name Data type	Description
4	Product Code integer 32 bit	This call returns the product code of the corresponding slave module. This value can be used, for example, to verify the bus structure.
8	Slave Online integer 32 bit	Indicates if a slave is online. This value includes the content of the AL status register of the respective slave. The value –1 indicates that currently, this value cannot be retrieved.
64	Name DataBuffer	This variable can be used to retrieve the module name of the respective slave. It must be accessed via the DLL function rdOptionBuf, where the buffer must have a size of at least 64 bytes and the name is returned as a zero-terminated string.

4.4.2 Reading EtherCAT master information

The structure of the connected EC bus is determined when the control system is started. It comprises, for example, the number of detected slaves and the expected and actual values for the Working Counters concerning input and output information.

4.4.2.1 Dev.# 900 hex

This device number can be used to read information on the states of the Working Counters. Write access is not possible here.

Index	SubIndex	Name Data type	Description
1	0	32 bit Integer	Expected Working Counter Domain 0 (Inputs)
1	1	32 bit Integer	Expected Working Counter Domain 1 (Outputs)
1	2	32 bit Integer	Expected Working Counter Domain 0+1 (Inputs + Outputs) Lower 16 bit = WC Inputs Higher 16 bit = WC Outputs
2	0	32 bit Integer	Expected Working Counter Domain 0 (Inputs)
2	1	32 bit Integer	Expected Working Counter Domain 1 (Outputs)
2	2	32 bit Integer	Expected Working Counter Domain 0+1 (Inputs + Outputs) Lower 16 bits = WC Inputs Higher 16 bits = WC Outputs
3	-	32 bit Integer	Number of detected slaves

The values of the Expected Working Counters are determined once and do not change anymore during the session. The current values of the Working Counters are adapted every time the bus structure changes. The expected values are not exceeded, though.
The number of the current slaves is always adapted. However, there might be a certain delay through scanning the bus structure.

4.4.3 Reading and writing SDOs of the bus subscribers

Slave subscribers with an SDO object interface permitting SDO access enable read and write access of these SDOs using the universal object interface. Access is possible from both SAP programming and PCAP programming. Relevant object descriptors (or AT variables) must be created by the user on the basis of the product documentation and the position of the respective slave modules in the EtherCAT bus.
The function returns an error if a subscriber does not allow SDO communication. In this case, the parameter value from read operations is invalid. An error is also returned if non-existent SDOs are accessed or access takes place with wrong data type assignment. Further information on the occurred error can be determined via dmesg in the fwsetup terminal screen.

When writing on SDOs, the written values are usually saved as resident on the slave module. Some modules provide a checksum which reflects changes of resident parameters. In these cases, the correct initialisation of the slave module can be checked.

Notice: SDOs must not be written on cyclically in programs because here, writing occurs in the flash memory of the EtherCAT slave modules. As the number of write operations is limited for this memory, the modules may otherwise be damaged irreparably after a short time.

The respective object descriptors are to be initialised as follows:

AccessType: Read or write
DataType: According to the data type indicated in the product documentation for the slave module
BusNumber: ECBus (600 dez.)
DeviceNumber: Index of the slave module in the EtherCAT bus (according to ethercat diag)
Index: SDO index according to the product documentation for the slave module
SubIndex: SDO subindex according to the product documentation for the slave module

In this way, revision numbers, for example, and other properties of EtherCAT slave modules can also be checked, in addition to the bus checking described in Chapter 4.4.1.

4.5 Manual reading of and writing on SDOs in the slave directory

This method is not to be confused with the mechanisms mentioned in Chapter 4.4.2. Here, it is described how the user can access internal variables of EtherCAT slave modules by operating the system manually during the setup phase. This operation occurs in the terminal window of the fwsetup program.

ethercat upload –p? 0xAAAA B

ethercat download –p? 0xAAAA B Wert

Option –t

For this, see also Chapter 6.3.

Explanations on the above commands:
-p? The ? stands for the index of the slave module in the EtherCAT bus, which is displayed, for example, via ethercat diag.
0xAAAA Hexadecimal object address
B Subindex of the object; here, too, it should be noted whether the information is decimal or hexadecimal. You can also enter hexadecimal numbers preceded by 0x.
Wert Numerical value to be written if necessary; this can be specified in decimal or hexadecimal form (preceded by 0x).

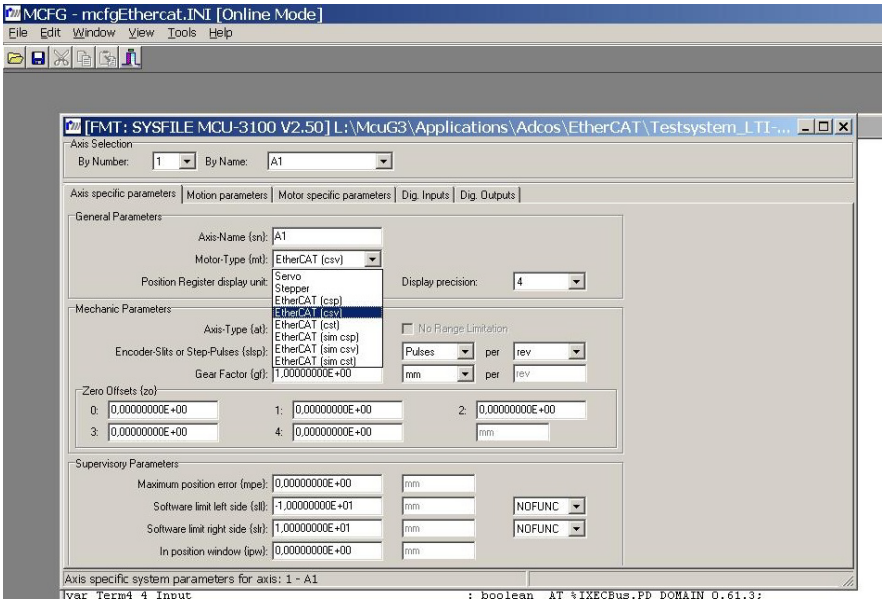
4.6 Commissioning of axes in mcfg.exe

4.6.1 Assignment of the motor axes

First, the axes of the system must be assigned to the existing axis types in the EtherCAT bus. This is done on the system data page, namely, in the "Motor Types {mt}" selection on the "Axis Specific Parameters" tab. For EtherCAT axes, the entries "EtherCAT (csp)", "EtherCAT (csv)" and "EtherCAT (cst)" are available here. The respective capability of the devices or the implementation in the operating system software of the APCI-8008 has to be considered for this.

The order of assignment must correspond to the order of the assigned axes in the bus, but must not be complete in the assignment. Thus, it is also possible to operate EtherCAT axes mixed with standard servo or stepper axes.

Figure 4-6: Assignment of an EtherCAT axis to the first axis in mcfg



4.6.2 Setting the actual value resolution

Also with EtherCAT axes, the resolution of the encoder system "Encoder Slits or Step Pulses {slsp}" must be found in the documentation or be read from the bus and entered here manually. The counter unit of {slsp} is usually "Pulses" because the possible electronic quadrupling always takes place in the EtherCAT module. The DS-402 profile contains the resolution of the encoder system in object 0x608F:1. This is a 32-bit integer variable.
According to Chapter 4.2, this variable can e.g. be read with the following command in the fwsetup monitor screen:

ethercat upload –p2 –tint32 0x608f 1

–p2 means that the variable is read from the slave with the index 2.
–tint32 means that the data type int32 is expected.

Notice: It is necessary to distinguish between the real resolution of the position encoder and the resolution provided by the EtherCAT slave. This generally does not have the same value. Precise information on how to determine the resolution must be provided by the manufacturer's documentation.

4.7 Further EtherCAT-specific handling of the system

4.7.1 Restart of the boards via BootFile

Since September 2021 (mcug3.dll V2.5.3.133), it is possible to restart the whole system using the DLL function BootFile (). The board is reset then like when booting the PC or with the reset button in fwsetup. But this step may only be carried out if no other programs are accessing the system.
The reboot process may take quite long. Work with the board can only be continued when InitmcuSystem? returns the value 0.

4.7.2 Transferring the dmesg protocol into the Windows file system

From mcug3.dll V2.5.3.133 and rwmos version V2.5.3.159, it is possible to read the dmesg protocol of the motion control board via DLL function. The protocol content on the board is thus rejected.
So, the user is provided with the Information which is otherwise displayed in the fwsetup monitor, with the dmesg command. This information can be used for the system diagnosis.

DLL function in C:

int rdDmesg (char *filename);

Parameter: File name with optional drive and path specification

Return values:

>= 0	For success; the number of bytes read is returned.
-1	Function in RWMOS not available
-33	RWMOS is no EtherCAT type.
-38	No motion control board detected in the system
-53	Error while opening the indicated file in Windows
-54	Internal error / Memory problem

Notice: The different lines are separated by LineFeed, not by CarriageReturn.

5 Drive-specific properties

For each drive system contained in the APCI-8008-EC, there are specific settings that must be considered when using a drive.
For example, there are no longer shared digital inputs and outputs. Instead, the digital inputs and outputs available on a drive are mapped in the digi or digo word of the respective axis. The assignment of the inputs/outputs to the single bits is documented below for each drive. Independently of that, all bits of the different EtherCAT registers can be accessed through the universal object interface as far as they are transferred cyclically via PDO (see Chapter 4.1.2.2.2).
Moreover, the Touch Probe functions which allow for position latch are configured and assigned to the latch functionality of the APCI-8008-EC. Thus, in general, a position latch is possible via digital input and via zero-trace signal as with previous boards.

Notice: Signal inversion / edge inversion with Touch Probe signals is not possible as before!

5.1 Omron servo drives

5.1.1 Accurax G5 drives

5.1.1.1 Operating modes

This device can be operated in the operating modes csv, csp and cst.

5.1.1.2 Digital inputs

Signal	Bit# digi word	

5.1.2 Digital outputs

Signal	Bit# digo word

5.1.3 Touch Probe function / Latch functionality

Touch Probe channel	Source / Trigger	Edge	Mode

5.1.4 Sysmac 1S servo drives

5.2 Panasonic MADHT1505

5.2.1 Operating modes

This device can be operated in the operating modes csv, csp and cst.

5.2.2 Digital inputs

Signal	Bit# digi word	
NOT	0	negative limit switch
POT	1	positive limit switch
HOME	2	home switch
SI-MON1	3	Latch Input EXT1
SI-MON2	4	General Purpose
SI-MON3	5	General Purpose
SI-MON4	6	General Purpose
SI-MON5	7	Emergency Stop E-STOP / General Purpose
INP	8	Positioning Complete

5.2.3 Digital outputs

Signal	Bit# digo word
EX-OUT1	0

5.2.4 Touch Probe function / Latch functionality

Touch Probe channel	Source / Trigger	Edge	Mode
1	EXT1	positive	Continous Triggering
2	Z-phase	positive	Trigger First Event

The flag NDX (Bit # 16) in the digi word cannot be used for this device because the EtherCAT bus does not provide any corresponding signal. For referencing, only the latched position can be used.

5.2.5 Quick Stop Deceleration (Object 6085 hex)

Here, a reasonable acceleration value must be entered in Counts / s2 via SDO access. This value must ensure that in case of error, the axis comes to a standstill after an adequate time. At a high encoder resolution, the default value of this object (1 000 000) is too low so that the axis may take too much time to stop.
This value becomes relevant when the EC bus is disconnected behind the respective slave during operation. The motion control board initialises a quick stop then (in csv and cst operation).

6 Configuration and diagnosis in the APCI-8008 console

6.1 Customisation of manufacturer-specific XML files for use with the APCI-8008

6.1.1 SM=??

This specification defines the SyncManager with which PDOs are transmitted cyclically. Registers that are read from the EtherCAT slave must be assigned to the SyncManager "3"; registers to be written to the EtherCAT slave, are assigned to the SyncManager "2".

6.1.2 Additions to PDOs

For this, exact knowledge of the manufacturer-specific object directory is necessary. Objects which are stored as resident on the slave must never be written cyclically; otherwise, the module would be damaged within a short time.

6.2 Restarting the EtherCAT module

Stop RWMOS with:
rwmoslkm stop

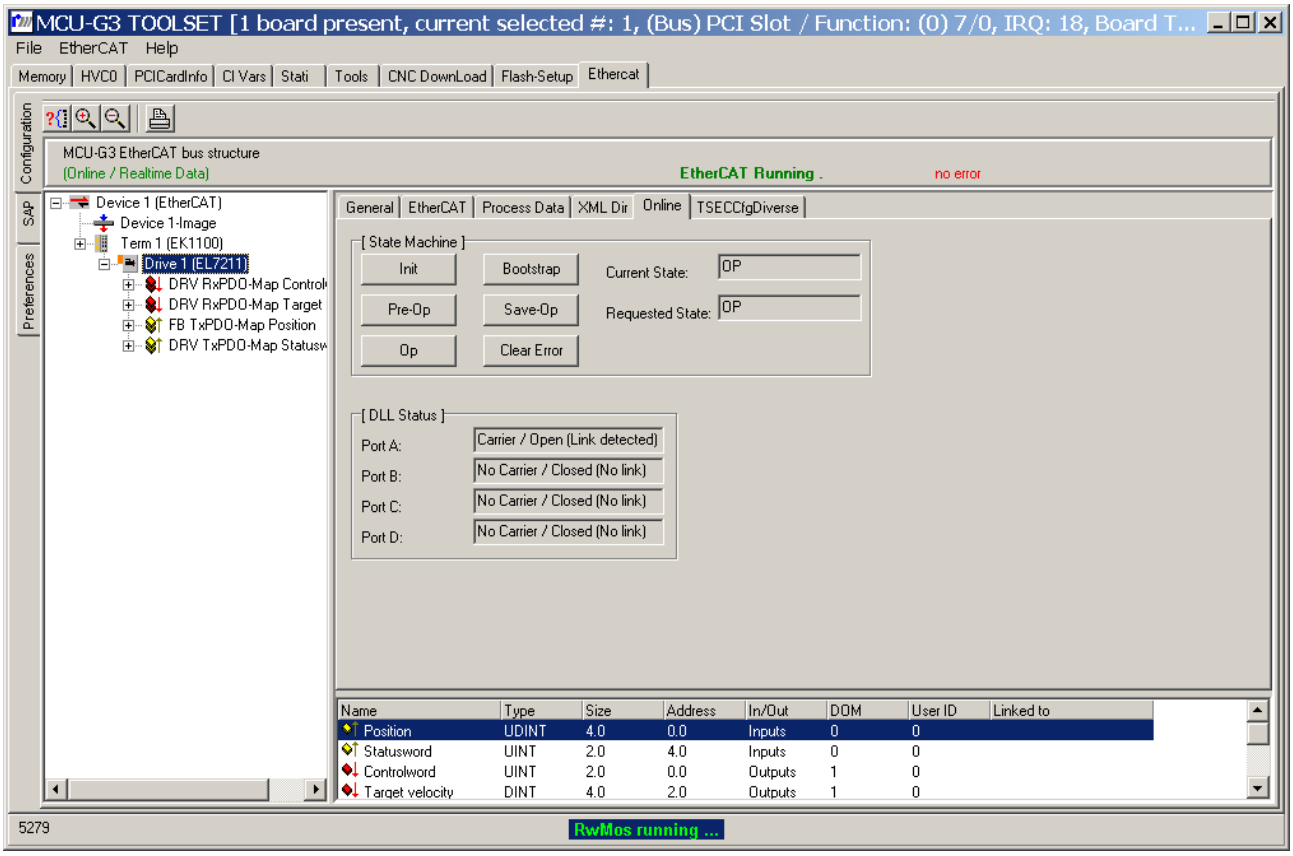
Restart of the EtherCAT module:
/etc/init.scrips/ethercat restart

Then RWMOS must be restarted:
rwmoslkm start

6.3 EtherCAT SDOs

The description of EtherCAT SDOs (Service Data Objects) of the respective slave module can be found in the corresponding product documentation. An SDO is defined by its index and subindex. Writing on SDOs may not be possible in the operational state of the slave module. In this case, the PreOperational State can be forced by clicking on the Pre-Op button in the shown view of fwsetup:

Figure 6-1: Pre-Op button



The RWMOS module of the motion control board can be stopped or restarted within Linux.

Starting and stopping the RWMOS module:

rwmoslkm stop

In this case, a restart is necessary afterwards, which can be initiated as follows:

rwmoslkm start

A hexadecimal index is represented below as 0xXXXX; a subindex as YY.

The reading of and writing on SDOs is performed with the following commands:

ethercat upload

and

ethercat download

Reading and displaying an SDO:

ethercat upload –p? 0xXXXX YY

The ? here represents the index of the slave component to be addressed.

Example for reading and displaying:

```
[root@MCU:~]# ethercat  upload  -p1  0x1c13  1
0x1a00  6656
```

The read value is output in hexadecimal and decimal form.

Example for writing on the SDO:

```
[root@MCU:~]# ethercat  download  -p1  0x1c13  1  0x1a00
```

Here, information is displayed only in case of error.

Optionally, sometimes a data type can or must be specified with the –t parameter.

Example:

```
[root@MCU:~]# ethercat  download  -p1  0x1c13  1  0x1a00 –tint32
```

A list of available data types can be displayed by calling the command without parameters:

ethercat upload

or

ethercat download

7 Appendix

7.1 Pitfalls when setting up the system

- One or more axes do not go into the operational state; the EtherCAT bus LED flashes.
  The reason may be that the EtherCAT axes defined in mcfg do not match the actual axes in the bus. The unassigned axes are then initialised as not valid.
  In this case, in the dmesg protocol, messages are displayed in the following form:
  EtherCAT WARNING 0-0: CoE Emergency Request received:
  Error code 0xFF1E, Error register 0x80, data
- When setting up the system data, attention must be paid to the setting in "Maximum / Minimum output Voltage" on the "Motor specific parameters" tab. Non-EtherCAT systems typically have a value of +/- 10 volts here. For EtherCAT systems, this value has a completely different meaning, e.g. in the unit of digits/second, and may take values of e.g. +/- 2e23. This corresponds to +8,388,607 and -8,388,608.
- Generating the SAP resource identifiers generates elements with unauthorised domains. It must be ensured here that all EtherCAT slaves are in the operational state when the SAP identifiers are generated.
- In the SAP variable declarations, the names of the AT specifiers contain peculiar, meaningless names, sometimes with the word "unknown".
  The reason for this is that in the supplied manufacturer's XML files not all of the PDOs required by the motion control board are written on. If necessary, a specially extended XML file must be used or an existing XML file be modified.
- At the start, the following error message is displayed in the fwsetup monitor screen:
  Failed to calculate Bus Topologie.
  The reason for this may be an error in bus cabling if, for example, a bus cable has not been connected to the EtherCAT input but to the EtherCAT output of a slave.

7.2 Implemented devices

7.2.1 I/O modules

Beckhoff	EL1014
Beckhoff	EL1804
Beckhoff	EL1809
Beckhoff	EL2004
Beckhoff	EL2202
Beckhoff	EL2809
Beckhoff	EL2828
Beckhoff	EL3064
Beckhoff	EL3102
Beckhoff	EL3255
Beckhoff	EL4002
Beckhoff	EL4132
Beckhoff	EL5101
Beckhoff	EL7211
Beckhoff	EK1814
Beckhoff	EP2316

ADDI-DATA	MSX-EC-1730-16
ADDI-DATA	CS-MAT-3001-16-EC

7.2.2 Drives

CopleyControl		AEP_180_18
CopleyControl		XEL_230_13
HIWIN		D1N
KOLLMORGEN		AKD
Lti		ServoJ
Omron	R88D	KN01H_ECT
		KN04H ECT
		KN08H ECT
		KN15F ECT
		KN20F ECT
		KN30F ECT
		1SN01H ECT
Rexrot IndraDriveCS		HSC01_1E_W0018

Beckhoff	EL7211

Panasonic	MADHT1505
Panasonic	MADHT1507
Panasonic	MADHT2510

Further devices may be added any time.

7.3 Figures

7.4 Tables