Operating Manual

PacDrive™ System

Article Nr.: 17130061-001
Edition: 2003-10

ELAU PACKAGING SOLUTIONS

Schneider Electric

ELAU AG
Dillberg 12
D-97828 Marktheidenfeld

Phone:	09391/606-0	Phone:	773 342 8400
Fax:	09391/606-300	Fax:	773 342 8404
eMail:	info@elau.de		sales@elau.com
Internet:	www.elau.de		www.elau.com

ELAU Inc.
4201 W. Wrightwood Ave.
Chicago, IL 60639

PD_UserMan_00_us.FM

Contents

PDM_UserMan_usIVZ.fm

PDM_UserMan_usIVZ.fm

PDM_UserMan_usIVZ.fm

1 On this manual

1.1 Introduction

Before using ELAU components for the first time, you should familiarize yourself with this operating manual.

In particular, observe the safety notes described in chapter 2.

Only persons who meet the criteria for "Selection and Qualification of Staff" (see chapter 2.4) are allowed to work on ELAU components.

One copy of this manual has to be available for staff working on the components with access at any time.

This manual is to help you use the component safely and expertly and to use it as directed.

Observe this manual. This will help to avoid risks, reduce repair costs and down times and increase the lifetime and reliability of the products.

You also need to observe the valid rules for the prevention of accidents and for environmental protection in the country and place where the device is used.

1.2 Symbols, Signs and Forms of Depiction

The following symbols and signs are used in this document:

Depiction	Meaning
■	First level enumeration sign.
–	Second level enumeration sign.
▶	**Action symbol**: The text following this symbol includes an instruction for action. Execute the instruction actions in the given order, from top to bottom.
✓	**Result symbol**: The text following this symbol contains the result of an action.
Italics	If the describing text contains special terms (e.g. parameters) these are written in italics.
Serif font	If the manual contains program code, this is marked by Serif font.
ℹ	**Information symbol**: This symbol marks notes and useful tips for using the product.
⚠	**Warning sign**: Safety notes can be found in the relevant places. They are marked by this symbol.
📖	After this symbol, information about contents of the chapter follows as guideline assistance.

Table 1-1: Symbols, signs and forms of depiction

2 General Safety Notes

This chapter contains general requirements for working safely. Every person using ELAU components or working on ELAU components has to read and observe these general safety notes.

If activities involve a residual risk, you will find a clear note in the respective places. The note describes the risk that may occur and preventive measures to avoid that risk.

2.1 Basics

The ELAU components are built according to the state of technology and generally accepted safety rules. Nevertheless, their use may cause a risk to life and limb or material damage if:

- you do not use the components as directed
- work on the components is not done by experts or instructed staff
- you inexpertly alter or modify a component
- you fail to test the protective measures in place after installation, commissioning or servicing
- you do not observe the safety notes and regulations.

Only operate the components in perfect technical condition, as directed, with regard to safety and risks and observe this manual.

The flawless and safe operation of the components requires appropriate transport, storage, mounting and installation as well as careful maintenance.

In case of any circumstances that impair the safety and cause changes in the operating behavior, immediately put the component(s) to a stop and inform the service staff in charge.

In addition to this manual, observe

- the prohibiting, warning and mandatory signs on the component, the connected components and in the switching cabinet
- the relevant laws and regulations
- the operating manuals of the other components
- the universally valid local and national rules for safety and the prevention of accidents.

2.2 Depiction of Safety Notes

Risk categories

The safety notes in this manual are grouped into different risk categories. The table below shows which risk and possible consequences the symbol (pictograph) and the signal words indicate.

Pictograph	Signal word	Definition
⚠	**DANGER!**	Indicates an immediately dangerous situation that will result in death or very serious injuries if the safety rules are not observed.
	WARNING!	Indicates a possibly dangerous situation that can result in serious injuries or major material damage if the safety rules are not observed.
	CAUTION!	Indicates a possibly dangerous situation that might result in material damage if the safety rules are not observed.

Table 2-1: Risk categories

2.3 Use as Directed

The ELAU components are designed for installation in a machine/ plant or for combination with other components to form a machine/ plant. The components may only be used under the installation and operating conditions described in this documentation. You must use the accessories and ancillary parts (components, cables, etc.) mentioned in the documentation. You must not use any foreign objects or components that are not explicitly approved by ELAU.

"Use as directed" also means that you

- observe the Operating Manuals and other documentations (see appendix),
- observe the instructions for inspection and maintenance.

Use other than directed

The operating conditions at the place where the device is used must be checked on the basis of the given technical data (performance information and ambient conditions) and observed.

The device must not be put into operation until it is guaranteed that the useable machine or the plant in which the motor is installed meets in its entirety EC directive 98/37/EC (machine directive).

In addition, observe the following norms, directives and regulations:

- DIN EN 60204 Safety of machines:
  Electrical equipment of machines.
- DIN EN 292 part 1 and part 2 Safety of machines:
  Basics, general design guidelines.
- DIN EN 50178 Equipment of high-voltage plants with electronic operating means.
- EMC directive 89/336/EEC

2.4 Selection and Qualification of Staff

This manual is aimed exclusively at technically qualified staff with detailed knowledge in the field of automation technology.

Only qualified staff can recognize the significance of safety notes and implement them accordingly.

This manual is aimed in particular at design and application engineers in the fields of mechanical and electrical engineering, at programmers, service and commissioning engineers.

Working on electrical equipment

Work on electrical equipment must only be done by qualified electricians or by instructed staff supervised by an electrician according to the electrotechnical rules.

An electrician is a person who, due to his vocational training, know-how and experience as well as knowledge of the valid regulations, is able to:

- evaluate the work he is supposed to do
- identify potential risks
- implement suitable safety measures.

2.5 Residual Risks

We minimized the health risk for people by means of appropriate construction and safety technology. Nevertheless, there is a residual risk, since the components work with electrical current and voltage.

2.5.1 Installation and Handling

WARNING!

Risk of injury while handling the unit!

Risk of injury due to squeezing, cutting or hitting!

- Observe the universally valid construction and safety rules for handling and installation.
- Use suitable installation and transport facilities and use them professionally. If necessary, use special tools.
- Take precautions against squeezing.
- If necessary, use suitable protective clothing (e.g. safety glasses, safety shoes, protective gloves).
- Do not stay under pending loads.
- Remove any leaking liquids from the floor immediately to avoid skidding.

2.5.2 Protection against Touching Electrical Parts

Touching parts carrying a voltage of 50 Volts or higher can be dangerous. When electric appliances are operated, certain parts of these appliances inevitably carry a dangerous voltage.

DANGER!

High voltage!

Life hazard!

- Observe the universally valid construction and safety rules for working on high-voltage units.
- After installation, check the fixed connection of the earth conductor on all electric appliances according to the connection plan.
- Operation, even for short-term measuring and test purposes, is only permitted with an earth conductor firmly connected to all electric components.
- Before accessing electrical parts with voltages exceeding 50 Volts, disconnect the unit from mains or power supply and lock it out. After switching off, wait for at lest 5 minutes before touching any components.
- Do not touch electrical connections of the components while the unit is on.
- Before switching on the unit, cover all voltage carrying parts to prevent accidental contact.
- Provide for protection against indirect touching (EN 50178 / 1998 section 5.3.2).

DANGER!

High leak current!

Life hazard!

- The leak current is greater than 3.5 mA. Therefore the units must have a firm connection to the power grid (according to DIN EN 50178 / 1998 - equipment of high-voltage systems).

PDM_SicherhMax_us_neu.fm

2.5.3 "Safely Separated Low Voltages"

PELV
Protective-Extra-Low-Voltage

Signal voltage and control voltage of the PacDrive units are <33 V. In this range the specification as PELV system according to IEC 364-4-41 includes a protective measure against directly and directly touching dangerous voltages by means of a "safe separation" from the primary to the secondary side in the plant/ machine. ELAU urgently recommends to execute the plant/ machine with safe separation.

DANGER!

High voltage due to wrong connection!

Life hazard or risk of serious injury!

- Only units, electric components or cables with a sufficient safe separation of the connected power supplies according to EN 50178 / 1998 (equipment of high-voltage systems with electronic operating means) may be connected to the signal voltage connections of these components.
- Make sure that the existing safe separation is retained throughout the entire current circuit.

FELV
Functional-Extra-Low-Voltage

When using ELAU components in systems that do not include a safe separation as a means of protection against directly or indirectly touching dangerous voltages, all connections and contacts (e.g. MAx-4, Sub-D connector, serial inerface) that do not comply with protection class IP2X must be permanently covered. The cover or device connection must be such that it can only be removed with the help of a tool. The protective measure must be observed on all connected devices.

2.5.4 Potentially Dangerous Movements

There can be different causes for potentially dangerous movements:

- mistakes in wiring or cable connection
- software errors
- faulty components
- errors in measuring value and signal encoders
- operating mistakes

The protection of people must be insured by superior means or monitoring on the plant side. You must not rely on the internal monitoring in the drive components alone. Monitoring or measures are to be provided according to a risk and error analysis by the plant builder according to the specific conditions of the plant. The valid safety rules for the plant are to be included in this process.

DANGER!

Potentially dangerous movements!

Life hazard, serious injury or material damage!

- No persons are allowed within the motion range of the machine. This is to be ensured by means of devices like protective fences, grids, covers or photoelectric barriers.
- The fences and covers must be sufficiently strong to withstand the maximum possible motion energy.
- The emergency stop switch must be located very close to the operator. Check the operation of the emergency stop before starting up the plant.
- Secure against unintentional start by enabling the mains contactor of the drives via an emergency off circuit or by means of the function 'safe stop'.
- Before accessing the danger zone, bring the drives to a safe stop.
- To work on the plant, power must be turned off and locked out.
- Avoid operating high-frequency, remote-control and radio devices in the vicinity of the plant's electronics and connecting wires. If the use of those devices is inevitable, check system and plant for possible malfunctions before first operation. In some cases a special EMT check may be necessary.

3 System Overview

3.1 Drive Concepts of Packaging Machines

Modern machine concepts in the packaging industry are characterized by the need for high dynamism, flexibility, modularity and efficiency. Packaging machines were traditionally equipped with a mechanical vertical shaft, which drove the secondary motions in the machine usually with mechanical components with complicated motion functions. Designing such a machine flexibly for different products is a highly complex task. Even minor changes in the packaging process, particularly in case of a product change, require major modifications and standstill time.

Packaging machines with electronic vertical shafts, by contrast, permit full flexibility. Electronic servo drive systems replace cam and coupling gears, and a virtual electronic vertical shaft ensures that the motion axes are synchronous. Any pulse and angle synchronous movements are determined by a central control.

Unplanned machine states, such as stop or emergency off situations or initialization movements can be realized synchronously. Dynamic changes of the goods to be packed or the packaging material in the plant, (like slippage of the products to be packed or expansion of the packaging material) can be registered by sensors while the machine is running and eliminated by modifying the corresponding drive movements. This development substantially changes and highly simplifies the classical mechanical machine concept. The structure of the packaging machine can be broken down into modules that are easy to apply and can be standardized.

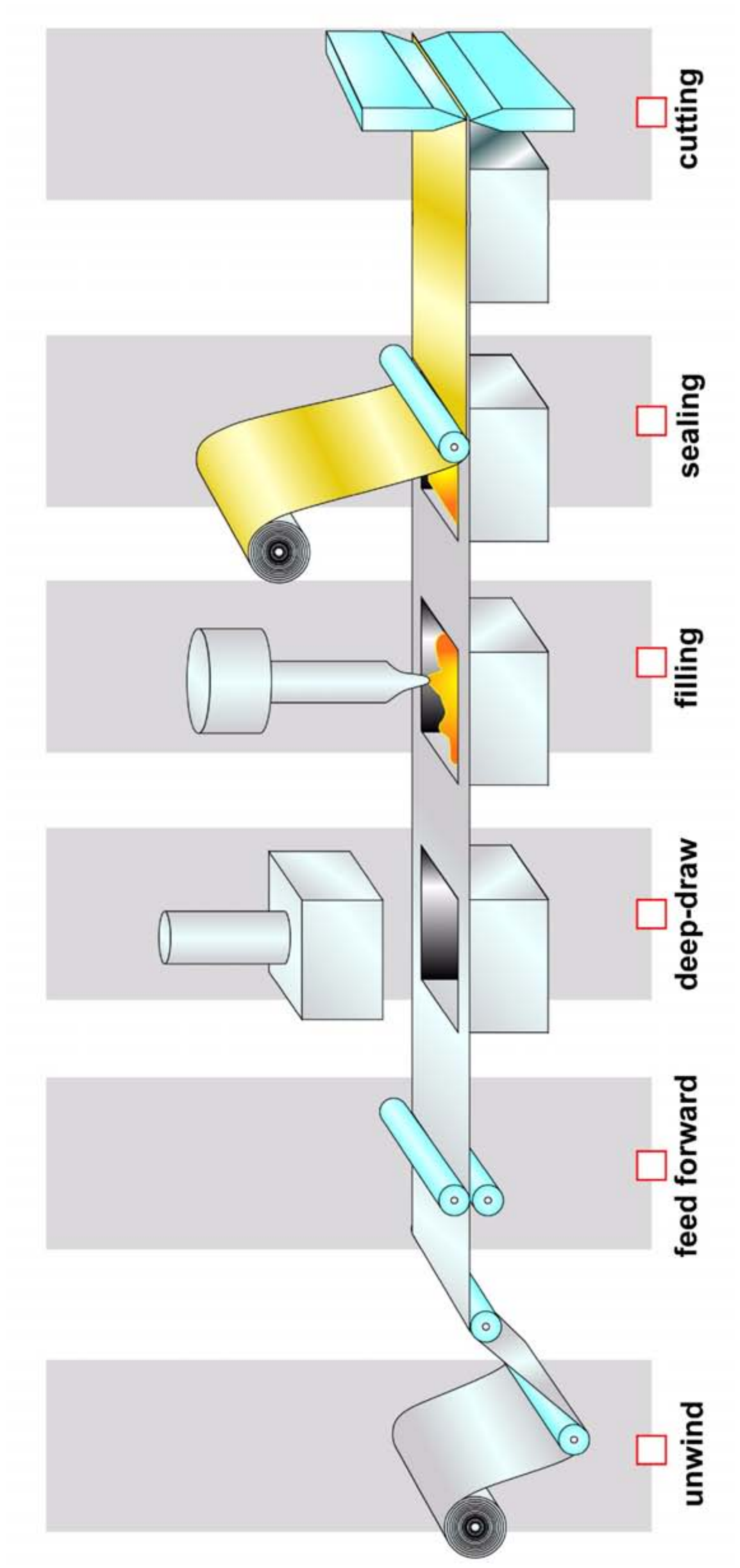


Fig. 3-1: Sketch of a packaging machine

3.2 Structure of the PacDrive™ Automation System

The PacDrive™ automation system offers a technically and economically optimal solution for electronic packaging machines. PacDrive™ consists of an efficient PC-based control, the MAx-4 PacController and the digital MC-4 MotorControllers, which include the mains connection unit, the end-stage power and the servo regulator of the individual axes (Fig. 3-2).

The MAx-4 PacController is the intelligent head of the system, and is based on an industrial PC. The MAx-4 PacController synchronizes and coordinates the motion functions of the packaging machine. Using an IEC 1131-3 soft PLC, it ventures into applications previously reserved for standard PLCs. The individual PLC or positioning tasks can be broken down into several parallel tasks, which are implemented with the EPAS-4 programming environment according to the IEC 1131-3 standard. Up to 40 servo axes can be connected to a MAx-4 PacController and supplied with positioning data.

The circular digital SERCOS real-time bus realizes the safe data exchange with the MC-4 MotorControllers. Due to the use of optical fiber technology, the data bus is insensitive to electromagnetic disturbance and cyclically supplies the decentralized MC-4 MotorControllers with new set values at a data rate of 4MBaud. All internal states of the axes can be checked via the real-time bus and processed in the MAx-4 PacController.

In addition to digital and analog inputs and outputs, the MAx-4 PacController has two serial interfaces and one Ethernet interface. A variety of process visualization and control systems can be connected to the PacDrive™ M via the integrated OPC interface. Further peripheral components can be connected via field bus interface modules. The MAx-4 PacController can act as a field bus master or slave.

The international field bus standards CANopen, PROFIBUS-DP and DeviceNet are supported. The built-in interfaces enable remote diagnosis via telephone modem or Internet. Via TCP/IP, PCs can communicate with the MAx-4 PacController and diagnose the state of the control directly.

3.3 Concept

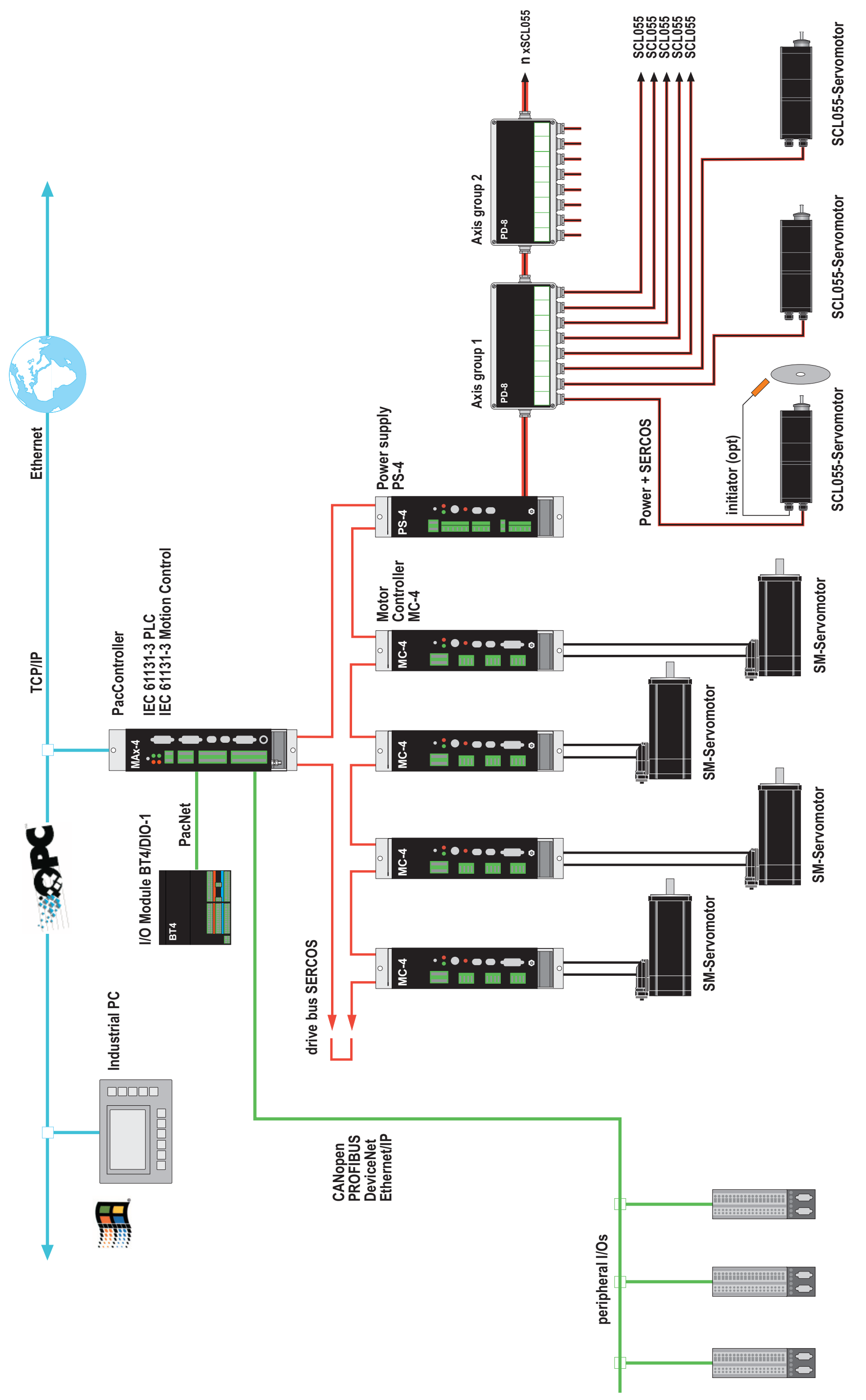


Fig. 3-2: System overview of the PacDrive™ automation concept

Alternatively, the PacController can be connected to a conventional PLC (superior to the PacController) via a field bus.

PDM_SysUebers_us.FM

3.4 Components

Automation Toolkit EPAS-4

EPAS-4 has extensive and proven functions and tools. A key advantage of EPAS-4 is that all components are integrated and intuitive.

For you as a user, this means:

Quick familiarization, eas handling, all tools integrated.

Libraries ELAU maintains extensive libraries geared to the packaging industry, helping you obtain cost effective and speedy answers to any concern. They will also help you to improve the quality of your user programs.

The highlights of EPAS-4 Automation Toolkit

- Runs under Windows (as of Win95)
- Programming language IEC 61131-3
- SCOPE tool (oscilloscope functions)
- Diagnosing tool
- very good debugging features
- serial or TCP/IP connection to the MAx-4 PacController

MAx-4 PacController

The MAx-4 PacController, a Pentium based controller hardware with VxWorks real-time operating system, realize the PLC and motion functions.

A PacController synchronizes, coordinates and generates the positioning functions for a maximum of 44 drives of a food and packaging machine.

For HMI tasks, various standard HMIs are used. Whether low-cost clear text or IPC - no problem for the flexible MAx-4.

The highlights of the MAx-4 PacController

- Pentium controller hardware
- VxWorks real-time operating system
- IEC 61131 PLC and motion control
- Coordinates up to 47 axis
- Standard interfaces to low cost operating panels or PC based HMI
- compact book size case
- Ethernet interface
- Standard field busses
- VarioCam® Motion Control
- SERCOS drive bus

MC-4 MotorController

Leading-edge technology

The digital MC-4 MotorController is characterised by its compact and autonomous structure suitable for wall mounting as well as its leading-edge technology. The innovative MC-4 has the mains supply unit, end stage and software regulator for one axis integrated in a compact casing. As it communicates with the PacController or PacPC only via fibre optical cable, it is also suitable for a decentralized structure. It requires no user program, comes with single- and multi-turn processing features as standard and configures itself with the help of the electronic name plate in the SM motor.

The highlights of the MC-4 MotorController

- World-wide voltage range
- Integrated mains supply unit
- Max. power 34,5 / 69 kVA
- Automatic motor recognition
- Minimum size
- Safety input Inverter Enable
- 250 % overload
- Integrated SERCOS interface
- few types

SM-Motor

Highly dynamic servo motors

Machines with fast cycle rates require highly dynamic AC servo motors. The SM motor series offers you as a user an optimum motor concept for your food and packaging machines. The dynamic brushless servo motors are furnished with high-resolution encoders (single-turn or multi-turn) and electronic name plate. Smooth surface and compact size meet the requirements of the target market.

The highlights of the SM motors

- Low mass moment of inertia
- 4-fold overload
- Reliable high-voltage technology
- Leading-edge magnetic technology
- High-resolution single- or multi-turn encoder
- Electronic name plate
- Plug in junction box
- IP 65 protection

4 The Basics of IEC-61131

4.1 Programmable Logic Controls

Around the end of the 1980s, a development started that made programmable logic controls (PLCs) central components of automation technology.

The PLC systems have various advantages compared with conventional relay technology: The systems can be adjusted fast in case of process changes, they are easily extendable etc. The function and structure of the systems are independent of the size of the PLC. A PLC consists of hardware and software. The hardware comprises a processor with storage components and further electronics for the connection of input and output systems. Modular PLC systems consist of various components such as component carriers with system bus, power unit, central processing unit and application storage, digital inputs and outputs and intelligent components for analog data processing or drive control.

For the input and output level, a standardized voltage of 24 V has been set. By means of the standardized voltage, a resistance that complies with the industrial environment was achieved.

The software comprises the operating system and the application program. The operating system manages the system resources and the organizational functions. In addition, the operating system ensures a controlled start after the operating voltage is switched on, it takes charge of error management and enables the exchange of information via communication groups. It coordinates the execution

of the application program, which maps the logic course of the control task.

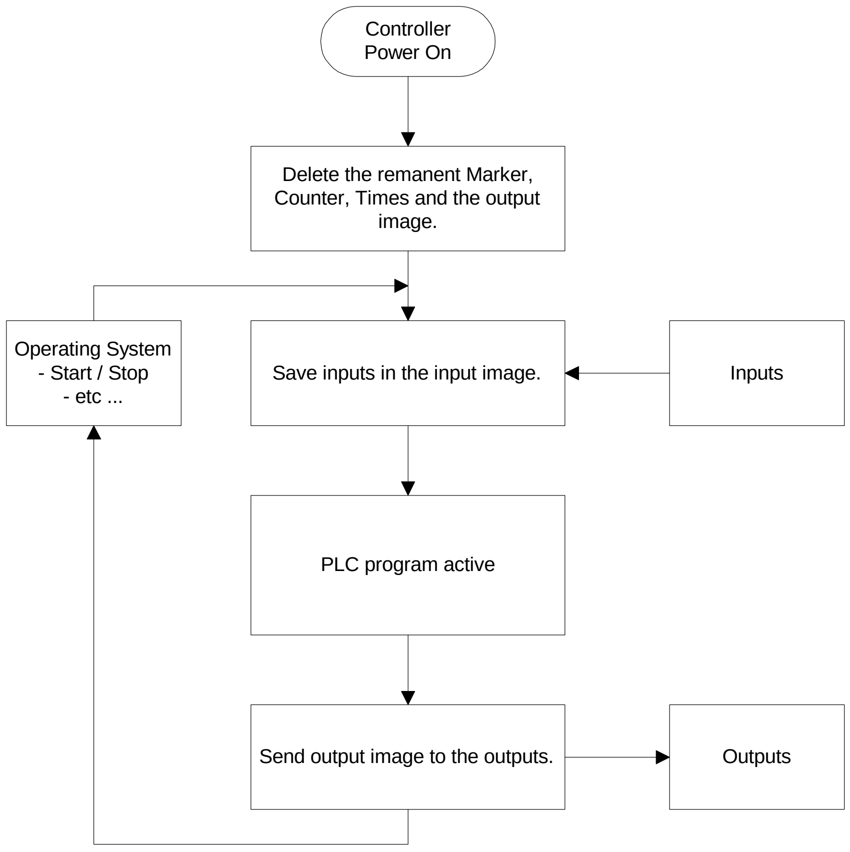

Controller
Power On
Delete the remanent Marker, Counter, Times and the output image.
Operating System
- Start / Stop
- etc ...
Save inputs in the input image.
Inputs
PLC program active
Send output image to the outputs.
Outputs

Fig. 4-1: Flow structure of a “classical“ PLC

An PLC starts processing the application program immediately after it is switched on (in the so-called RUN mode). Directly after the system is switched on, all non-remanent memories are reset and the processing cycle starts scanning in the input image.Then, the complete PLC instruction list is processed. In the next step, the output image that was created through the instruction list is sent to the outputs. The program cycle starts anew with scanning in the input image. The cycle time is probably the most important speed criterion for a PLC. In practical operation, it typically ranges from some ms to several hundred ms.

NOTE

The sequence structure described here applies in particular to single-task systems. Since the PacDrive^{TM} system is a multi-task system, there are deviations.
See also „Special Features of the PacDrive ^{TM} “, page 86.

The simple flow structure shows its advantages when data and programs are manipulated in ongoing operation. Almost all programming environments allow for the change of all variables during a processing cycle. Due to the set program structure, program sequences can be loaded at run time (Online Change), since there are always exact starting points at the beginning and the end of a cycle.

When these new systems were launched, the producers had to make their customers, who were used to contactors, acquainted with the programming of the new systems. For that reason, numerous but mostly short-lived attemps were made to find a programming language that facilitates the change from the traditional detailed wiring diagram to the PLC program. The programming languages Function Block Diagram, Ladder Diagram and Instruction List were developed then.

At that time, however, developers failed to define a universally binding standard for programming PLC systems. In consequence, each producer had specific characteristics in the programming languages Function Block Diagram, Ladder Diagram and Instruction List. Therefore, company standards emerged in various geographic regions, such as Siemens STEP 5 in Europe, Allen Bradley in the U.S. and MITSUBISHI in the Far East.

4.2 IEC-61131

In recent years, the complexity of the applications and thus the programming costs grew at a superproportional rate. The producer-specific standards did not ensure that the programs could be re-used.

Due to the above reasons, the so-called IEC standard *IEC 61131* was worked out at the beginning of the 1990s under the lead of the *International Electrotechnical Commission (IEC)*. Within the context of this process, the new languages *Structured Text* (ST) and *Sequential Function Chart* (SFC) were defined in addition to the existing languages *Ladder Diagram* (LD), *Function Block Diagram* (FBD) and *Instruction List* (IL). IEC 61131 sums up the requirements for a modern PLC system. The standard was not intended as a rigid specification, but as a guideline for PLC programming. Accordingly, the standard describes the significant properties of an PLC while leaving producers enough room to use their own implementation.

IEC-61131 tried to bring modern software engineering into the PLC world. This was necessary due to more complex processes and functions and the cost explosion regarding the development of application programs.

Tested and standardized software modules which can be re-used was regarded as a solution. However, this is made more complicated due to:

- direct addressing
- untypisized variables,
- no type check.

In general, the objectives of IEC 61131 are:

- application of software engineering methods with the aim of reusable software modules
- holistic approach to problem solutions
- abstraction of complex tasks into smaller modules
- definition of unambiguous interfaces
- standardization of the language scope in order to increase portability.

4.2.1 The Programming Model

In addition to the elements for the programming and organization of the application program, IEC 61131 also gives guidelines for modelling and structuring PLC composite systems. To structure the system, the concepts of configuration and resource were introduced.

The model also takes into consideration properties such as multi-processor systems, modern PLC operating systems featuring multi-tasking properties, unlimited number of analog and digital inputs and outputs and the ability to communicate with other PLCs and computers.

A configuration defines the structure of a system. For example, this can be a PLC with several, even networked, CPUs on machine cell level. A configuration is comprised of one or more resources, which represent sub-controls with own signal processing functions. In a real configuration, a resource is represented by a PLC CPU, which in most cases has multi-tasking capabilities.

A resource is structured with the help of one or more programs which are controlled via tasks. A task is an executable program unit to which both a priority and an execution type are assigned. There-fore, execution lines with different properties can be formulated within one and the same program. Thus, not only cyclic taks with a system-wide uniform cycle time are possible. Cycle times can also be combined and event-controlled program units can be made available in the system. By assigning a certain priority to a task, CPU time is allocated to a resource.
The run time properties of the complete program, which can run indepentently in a CPU, are defined by linking programs to a certain task. Thanks to the flexibility of the system modulation, a program can be linked to several tasks, thus creating several instances with different run time properties.

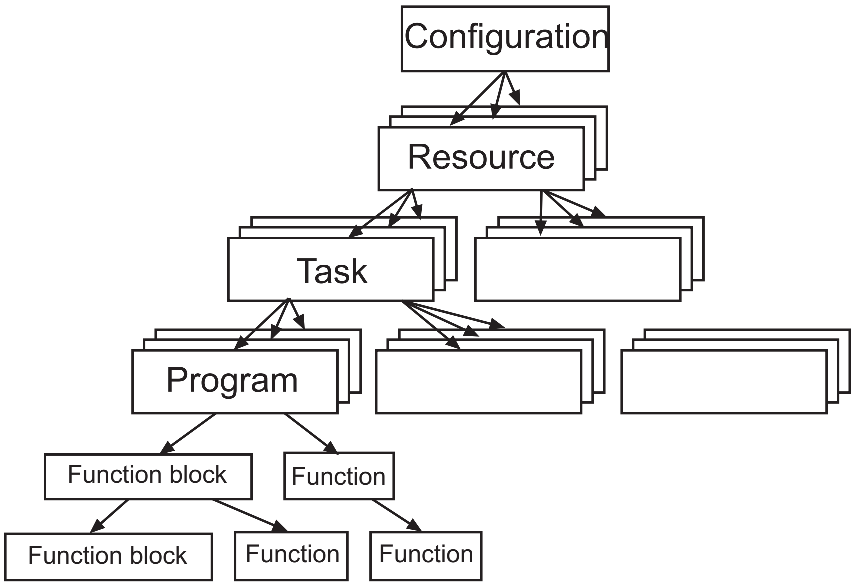


Fig. 4-2: The programming model according to IEC 61131

IEC 61131-3 supports local data that can be declared in programs, function blocks or functions. Local data can only be accessed in the corresponding program hierarchy level and represent a mechanism for data encapsulation. Of course also resource-wide available global data available for all program elements are possible. If multitasking systems are used, however, access to global data is a risk source for inconsistent data. In addition, there are directly accessible data with fixed addresses within the PLC address range. Usually, those are the addresses of the inputs (I), outputs (O) and markers (M, also called flags).
Another aspect of the programming model describes the re-booting behavior of the control. Both the cold restart and the hot restart (reset) are described. With a cold restart, the program is loaded anew. All variables are set to their initial values. Either a default initial value or a value defined by the programmer is set. All tasks of the resource are started. In case of a hot restart (reset), by contrast, the variables are not set to their initial values. Instead, the values held in the storage prior to the interruption will be taken over.

4.2.2 The Communication Model

An essential aspect in the description of the structural elements is the data exchange. With the help of the communication model defined in the IEC 61131, it is possible to create well structured and in particular modularized PLC programs, which is a fundamental basic characteristic for the development of application-oriented, reusable program modules. In priniciple, IEC 61131 provides for the following communication possibilities:

- access paths (VAR_ACCESS)
- global variables (VAR_GLOBAL, VAR_EXTERNAL)
- call parameters
- communication modules (IEC 61131-5)

All elements of a configuration communicate with one another and also with other computer systems exclusively via defined a*ccess paths*. In addition, *global variables* are used for the simple communication of programs within a configuration. Global variables can be placed and used on configuration, resource and program level.
The data exchange within programs is effected via *call parameters*, input and output variables or function values. Although high-level language programmers are well acquainted with this structural tool, it brought fundamentally new aspects into conventional PLC programming. Call parameters and transfer variables allow for the definition of unambiguous interfaces and thus make an important contribution to the encapsulation of functionality.
In addition to the elements of the communication model described so far, also special „communication modules“ can be used. They are of monolithic nature and are linked into a program. Thanks to those modules, data exchange between sender and receiver is self-sufficient. Communication services are defined in part 5 of IEC 1131, which is still being edited, however.
Looking at the communication model of IEC 1131, the good support of standardized software modules is particularly striking. Thanks to the encapsulation of functionality and data, a clearly defined interface and a side-effect free behavior, the acceptance of the modules among users has increased significantly.

An important standardization of programming languages according to IEC 61131-3 is the definition of data types. The norm includes various elementary data types from which derived and user-defined data types can be composed.

The user can use standard data types and self-defined data types for programming. Assigned to each identifier is a data type that defines how much memory space is reserved and which values correspond to the memory content.

Elementary Data Types

IEC 1131-3 defines five groups of elementary data types. Their related general data type is given in brackets.

- Bit string (ANY_BIT)
- Integer with and without algebraic sign (ANY_INT)
- Floating point (ANY_REAL)
- Date, time (ANY_DATE)
- String, duration, derived (ANY)

Data type	Description	No. of bits	Range	Initial
BOOL	boolean	1	[0, 1]	0
BYTE	bit string 8	8	[0, ..., 255]	0
WORD	bit string 15	16	[0, ..., 65535]	0
DWORD	bit string 32	32	[0, ..., 4,295 E09]	0
LWORD	bit string 64	64	[0, ..., 1,845 E19]	0

Table 4-1: Bit string

Data type	Description	No. of bits	Range	Initial
SINT	short integer	8	[-128, ..., +127]	0
INT	integer	16	[-32768, ..., +32767]	0
DINT	double integer	32	[-2^{31}, ..., +2^{31}-1]	0
LINT	long integer	64	[-2^{63}, ..., +2^{63}-1]	0
USINT	short integer	8	[0, ..., +255]	0
UINT	integer	16	[0, ..., +65535]	0
UDINT	double integer	32	[0, ..., +2^{32}-1]	0
ULINT	long integer	64	[0, ..., +2^{64}-1]	0

Table 4-2: Integer with and without sign

Data type	Description	No. of bits	Range	Initial
REAL	floating-point number	32	see IEC 559	0.0
LREAL	long floating-point number	64	see IEC 559	0.0

Table 4-3: Floating point

Data type	Description	No. of bits	Range	Initial
DATE	date	like DWORD		0001-01-01
TOD	time	like DWORD		00:00:00
DT	date and time	like DWORD		0001-01-01-00:00:00

Table 4-4: Date, time

Data type	Description	No. of bits	Range	Initial
TIME	duration	like DWORD		0 s
STRING	string	Default=80		"empty" string

Table 4-5: Date, time

Example of a string declaration:
str:STRING(35):='This is a string'

NOTE

The PacDrive™ M system does not support the LWORD, LINT and ULINT data types.

Data types - notation

The data types listed above can be represented in various forms:

BOOL, BYTE, WORD, DWORD, LWORD

PD_UserMan_IEC_us.fm

These data types can be represented as follows:

- TRUE or 1
- FALSE or 0
- decimal, hexadecimal (16#), octal (8#) or binary (2#) representation

Example for WORD: 234, 16#ff, 2#1001_1100_0011_1111

<u>SINT, INT, DINT, LINT, USINT, UINT, UDINT, ULINT</u>

Decimal, hexadecimal (16#), octal (8#) or binary (2#) representation. The underscore (_) separates units.

Examples:

Decimal representation for INT: -123, +234, 0, 1_000
Hexadecimal representation for INT: 16#F1, 16#0A_1B
Binary representation for INT: 2#0001_0011_0111_1111

<u>REAL, LREAL</u>

Normal decimal representation with decimal point or exponential representation.

Example: 1000.23 and 1.23e3 and 1.23E3 and 1.23E03 are interpreted identically.

<u>TIME (Time duration)</u>

TIME#, t# or T# stand at the beginning of a time/date designation. Overflow is allowed (e.g. 25 hours).
d stands for days, h for hours, m for minutes, s for seconds and ms for milliseconds. The underscore (_) separates units.

Example: T#2d_26h_4m_12s_123ms

<u>DATE, TIME_OF_DAY or TOD, DATE_AND_TIME or DT</u>

DATE# or D# stands for a date
TIME_OF_DAY# or TOD# stands for a time of day
DATE_AND_TIME# or DT# stands for time of day and date
Date: D#1998-12-07 stands for July 7, 1998
Time of day notation: TOD#12:00:00.123
Date and time: 1998-12-07-12:00:00.123

<u>STRING</u>

Inverted commas ‘ ‘ contain a string.
The dollar sign $ leads control characters (line feed, tabs).

Examples:
String, Control charcter: ‘This is a line feed character $L‘
Empty String:

Derived data types

According to IEC 1131-3, data types can be derived from the elementary data types. Data types that can be derived are:

- Arrays (ARRAY)

- Pointers (POINTER)
- Enumeration types
- Structures (STRUCT)
- References

With the help of derived data types, complex structures can be managed with simple constructs: With just one assignment, complex data can be transmitted to functions and function blocks. The derived data types can be composed from the basic data types and derived data types.

Arrays (ARRAY)

An array is composed of several individual variables of the same type.
One-, two- and three-dimensional arrays are supported by elementary data types. Arrays can be defined in the declaration part of a block and in the global variable lists.

Syntax:

<array_name>: **ARRAY** [<ug1>..<og1>,<ug2>..<og2>] **OF** <elem. type>.

ug1, ug2 represent the lower limit of the array range and og1, og2 the upper limit. The limiting values have to be integer.

Example:

Card game: **ARRAY** [1..13, 1..4] **OF** INT;

Initialization of arrays:
Either all elements of an array are initialized or none of them.

Examples for initializations of arrays:

```
arr1 : ARRAY [1..5] OF INT := 1,2,3,4,5;
arr2 : ARRAY [1..2,3..4] OF INT := 1,3(7);
                (* short for 1,7,7,7 *)
arr3 : ARRAY [1..2,2..3,3..4] OF INT := 2(0),4(4),2,3;
                (* short for 0,0,4,4,4,4,2,3 *)
```

In a two-dimensional array, array components are accessed with the following syntax:

<array_name>[Index1,Index2]

Example:

```
CardGame[9,2]
```

PD_UserMan_IEC_us.fm

Pointer (POINTER)

NOTE

You should avoid using pointers, since they cannot be checked by the compiler and at run time.
In most cases, the parameter transfer with VAR_IN_OUT is the better solution.

Pointers feature a data type and an address. The data type is the data type of the data elements to which the pointer is pointing. The address is the address (the position) where the data element is stored in the memory unit.

In pointers, the address of variables or function blocks is stored during the run time of a program.

Pointer declarations have the following syntax:

<identifier>: **POINTER TO** <data type/function block>;

A pointer can point to any data type and function block, including self-defined ones.

With the address operator ADR, the address of a variable or function block is assigned to the pointer.

The dereferentiation of a pointer is effected via the content operator "^" after the pointer identifier.

Example:

```
pt:POINTER TO INT;
var_int1:INT := 5;
var_int2:INT;
pt := ADR(var_int1);
var_int2:= pt^; (* var_int2 is now 5 *)
```

Enumeration type

An enumeration type is a self-defined data type which is made up of a number of self-defined string constants. Those constants are called enumeration values.

The enumeration values are known throughout the entire project, even if they were declared locally in a block. They start with the keyword TYPE and end with END_TYPE.

Syntax:

TYPE <identifier>:(<Enum_0> ,<Enum_1>, ...,<Enum_n>);

END_TYPE

The <identifier> can take on one of the enumeration values and will be initialized with the first one. The values are compatible to full numbers, i.e., they can be used to carry out operations as with INT.

A number x can be assigned to the <identifier>. If the enumeration values are not initialized, the enumeration begins with 0. In the initialization process, make sure that initial values are ascending. The validity of the number will be checked at run time.

Example:

```
TrafficLights: (red, yellow, green:=10); (*red has the
  initial value 0, yellow 1, green 10 *)
TrafficLights:=0; (* Traffic lights have the value
  red*)
FOR i:= red TO green DO
  i := i + 1;
END_FOR;
```

The same enumeration value must not be used twice.

Example:

```
TrafficLights: (red, yellow, green);
Color: (blue, white, red);
```

Error: red must not be used for both TRAFFIC LIGHTS and COLOR.

Structures (STRUCT)

A structure is composed of several individual data elements that belong together from the user's point of view. The individual elements can belong to different data types.

Structures start with the keyword TYPE and end with END_TYPE.

Structure declarations have the following syntax:

TYPE <structure name>:

STRUCT

    <variable declaration 1>
    .
    .
    <variable declaration n>

END_STRUCT

END_TYPE

Nested structures are allowed. The only limitation is that variables cannot be put on addresses. (AT declaration is not allowed!)

Example for a structure definition named Polygonzug:

```
TYPE Polygonzug:
STRUCT
  Start:ARRAY [1..2] OF INT;
  Point1:ARRAY [1..2] OF INT;
```

```
   Point2:ARRAY [1..2] OF INT;
   Point3:ARRAY [1..2] OF INT;
   Point4:ARRAY [1..2] OF INT;
   End:ARRAY [1..2] OF INT;
END_STRUCT
END_TYPE
```

Components of structures are accessed with the following syntax:

<structure_name>.<component name>

If there is e.g. a structure named „week“, which includes a component named „Monday“, it is accessed in the following way:

Week.Monday

References

The self-defined data type reference is used to create an alternative name for a variable, constant or function block.

References start with the keyword TYPE and end with END_TYPE.

Syntax:

TYPE <identifier>: <assignment expression>;

END_TYPE

Example:

```
TYPE message:STRING[50];
END_TYPE;
```

Declaration of variables and constants

Variables are identifiers defined by the user. They are used as wild cards for the data of the PLC program. A main characteritsic of variables is that their content can be changed. Under IEC 61131-3, variables are used for storing and processing information. Variables are identifiers defined by the user.
IEC 61131-3 includes five different categories of variables:

- Global variables
- Local variables
- Input variables
- Output variables
- Input and output variables.

Input, output and input/output variables are related to a program, a function or a function block. They can be used in the specific way only. Within the assigned program organization unit (POU), they can be changed in both reading and writing manners, outside the assigned POU, only in the defined manner.

According to IEC 61131-3, a marker is a sequence of characters, numbers and underscores (_). The sequence has to start either with a letter or an underscore. Markers must not contain blank spaces, special characters and ä, ö, ü. 32 significant characters are distinguished. Case sensitivity is not recognized.

The variables defined by the user have to be declared (explained). All variables used in a POU have to be explained in the declaration part of the POU.

The declaration part must contain one of the following keywords in text form:

- VAR
- VAR_INPUT
- VAR_OUTPUT

Declarations are separated by a semicolon and end with the keyword VAR_END.

Keyword	Use of the variable
VAR	within the POU
VAR_INPUT	coming from outside; cannot be changed within the POU
VAR_OUTPUT	supplied from the POU to the outside
VAR_IN_OUT	coming from outside; can be changed within the POU
VAR_EXTERNAL	supplied by the configuration; can be changed within the POU
VAR_GLOBAL	declaration of global variables

Table 4-6: Keywords for variable declaration

Variable attributes

Keyword	Use of the variable
RETAIN	variable is buffered (zero-voltage proof)
CONSTANT	constant “variable“; cannot be changed
AT	allocation of memory place

Table 4-7: Keywords for variable declaration / attributes

Priority	Allocation	Start behavior
1 (highest)	zero-voltage proof by RETAIN (battery-buffered)	*hot restart*: restoration if voltage returns or after stop
2	start value set by declaration	*cold restart*: start values for defined new start
3 (lowest)	start value for data type (predefined)	*cold restart*: start values for defined new start

Table 4-8: Allocation of start value types by priority

NOTE

New initial values can be assigned to derived data types.

Variables have various scopes of validity

They can be declared outside a POU and used program-wide, included into the POU as call parameters or be of local importance for the POU.
If a variable is declared in a function block, it is valid for this very function block only. If it is declared in a program, it is valid for all function blocks declared in this program.

<u>Variables</u>
<name>:<TYP>;
Example: Test: BOOL;

<u>Constant expressions</u>
VAR CONSTANT <name>:<TYP>:=<value>
Example: VAR CONSTANT Test: BOOL:=TRUE;

Localized - firmly addressed - variables

Such variables resemble conventional PLC technology. They are of certain importance in IEC 61131-3 systems since they can be used to link two properties:

- All process signals are linked via localized variables.
- Overlappings of localized variables are allowed and can be used as a programming tool.

- The definition of a certain memory space for variables starts with the AT identifier and defines three parameters:
- the start address (after the AT identifier as offset of zero)
- the memory range (input, output, marker)
- the length of the variables in the memory (by means of a variable type abbreviation)

In this case,

- the first letter of the length definition is a „%“
- the second letter is
  - I for input , Q for output, M for marker
  - the length of a variable in the memory (by means of variable type abbreviation)
- the third letter defines the length:
  - X for bits (the bit address is always complemented in „Byte.N“, e.g. 1.0),
  - B for byte,
  - W for word,
  - D for double and
  - L for long word.

Examples:

```
%IB24, %QX1.1, %MW12
```

Overlapping location of variables

Overlapping locations of variables are allowed, e.g. %MB12 is the first byte of %MW12 and the first byte of %MD12. Also on bit level, this kind of overlapping is desired and to be used in a meaningful way: %MX12.0 is the first (least significant) bit of %MB12.

Initialization of variables

In principle, each variable is initialized after a cold restart. Usually, the default value is 0 or FALSE. A user-specific setting to another value is of course possible. In the declaration, it is assigned with the = sign.
The initialization can also be effected within derived data types (arrays, structures). The necessary syntax is shown in the example.

Example:

```
VAR
a : INT := 13;
b : STRING := ‘this is a string‘;
c : REAL := 1.1;
END_VAR
```

```
VAR
a : myStruct :=
  (
  state := TRUE,
  inputValue := 2.5
  );
END_VAR

VAR
a : ARRAY[1..10] OF INT :=
  1, 2, 2(4), 5, 6, 7, 8, 9,10;
END_VAR
```

4.2.3 Program Organization Units POU

IEC 61131-3 limited the diversity of module types used in existing producer-specific PLC programming models. With so-called program organization units (POUs), the program organization is made easier and more homogenous.
In earlier systems, the implicit, non-transparent and producer-dependent meaning of individual modules or module areas had made it difficult if not impossible even for experienced program developers to change to a PLC of another producer.
By structuring a POU into *program*, *function* and *function block*, a comprehensible depth was chosen, making it possible to manage the complete application-specific implementation.

Program

This POU type represents the “main program“. All variables of the complete program to which physical addresses are assigned (such as inputs and outputs of the PLC) have to be declared in this POU or on a higher level (*resource*, *configuration*). Access to global variables and access paths.

Function

A *function* describes a complex linking logic that has no „memory“, i.e. no static variables. A special feature of a function is that it will always return the same result if the input values are the same.

Example:

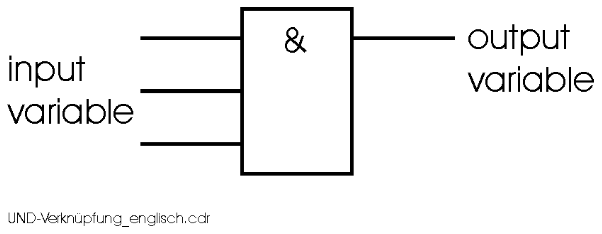


Fig. 4-3: The AND connection is a boolean standard funciton

Function block

If an intelligent module with a memory is needed, the *function block* (FB) with local static variables offers the necessary basis. An FB (e.g. time or counter) can return different results even if the input data are the same (e.g. regarding a time or counter). According to the new norm, each instance of an FB has its own „encapsulated“ data range where the calculations are done: the instance (see below).
In order to standardize typical PLC functionalities, the new standard introduced standard functions and function blocks. This *library* is an important basis for uniform, producer-independent programming of PLC systems.

PD_UserMan_IEC_us.fm

Example:
The switch-on delay TON (Timer ON) is a function block (standard FB). If the Boolean variable „1“ is created at INPUT (IN), the link time PT (time value) is started. If the time ran out, output Q takes on the Boolean variable „1“. At output ET (elapsed time) the present actual value of the time can be read.

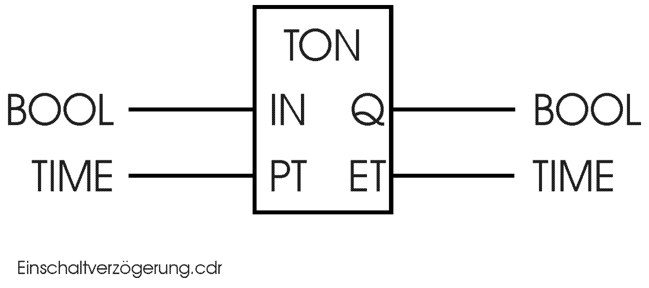


Fig. 4-4: Switch-on delay TON (Timer ON)

In summary, a POU can be regarded as a unit that can be transla- ted by the compiler of a PLC programming environment independently of other program parts. The properties of a POU allow for a wide-reaching modularization of the application program and the re-use of implemented and tested software modules. In order to enable program modules to access POUs, at least the declaration of the call interface is required (prototype). Translated program parts can later be linked into a joint program (linker). In contrast to some high-level languages, IEC 61131 knows no validity range for POUs. In a project, the name of a POU is global and cannot be assigned more than once. After its declaration, a POU is globally available to all other POUs.

Structure of program organization units

POUs consist of a declaration part and an instruction part.
In the declaration part, the variables are declared.
Following the declaration part, the instruction part describes the instructions that are to be executed.

Keyword	POU	Comment
PROGRAM	program	This is a main program, which enables access to the PLC periphery. Global variables and access paths are defined here.
FUNCTION_BLOCK	function block	Has input and output variables; is used very often for program creation.
FUNCTION	function	Simple PLC component which expands the stock of operations in programing.
END_FUNCTION		End mark of the POU.

Table 4-9: Program organization units with keywords

<u>Functions</u>

Functions can be coded in all IEC 61131-3 languages except SFC. The return value is declared with the name of the function.

Example:

```
FUNCTION TEST: REAL
(* -> Data type of the return value is REAL *)
(* Declaration part*)
VAR_INPUT
  Switch1: BOOL;
  Switch2: BOOL;
END_VAR
VAR
  Intermediate result: BOOL;
END_VAR
(* Instruction part*)
TEST:= Switch1 AND Switch2;
(* -> Assign return value to function *)
END_FUNCTION
```

NOTE

IEC 61131-3 defines standard functions, which are supported by most implementations. These standard functions are explained in detail in the Programming Manual -Reference-.

Function Blocks

Function blocks are used to set input, output and internal variables. States of an FB call are buffered from cycle to cycle. In that process, the program code of the FB creates changes of the input, output and internal variables.
Only the input and output variables can be reached from the calling program. Calls from other FBs are allowed, from all languages into all languages.

Example:

```
FUNCTION_BLOCK Counter
(* Declaration part*)
VAR_INPUT
   iMode : INT; (* 0 = Reset, 1 = Count *)
END_VAR
VAR_OUTPUT
   iCounter : INT; (*actual counter value*)
END_VAR
(* Instruction part*)
IF iMode = 0 THEN
   iCounter:= 0; (*Reset*)
ELSEIF Mode = 1 THEN
   iCounter:= iCounter + 1;
ENDIF;
END_FUNCTION_BLOCK
```

NOTE

IEC 61131-3 defines standard function blocks which are supported by most implementations. These standard function blocks are explained in detail in the Programming Manual -Reference-.

IEC 61131-3 provides for the instancing of function blocks: An instance is a structure in which all internal variables, inputs and outputs of a call of an FB are stored. In consequence, a program which calls FB1 five times has five instances of FB1, one for each call.
The advantage of this rather uncommon *object-oriented* procedure is that the program diagnosis can be made exactly to the call and without side effects. With an automatic declaration, modern tools help carry out this instancing: For a call of an FB, an instance name which manages the data of this call is set.
An important characteristic is that all instances use the same program code of the FB. For that reason, changes in the program code have the same consequences in all calls. Thus, an instance is no copy of the FB for a call.

Example:

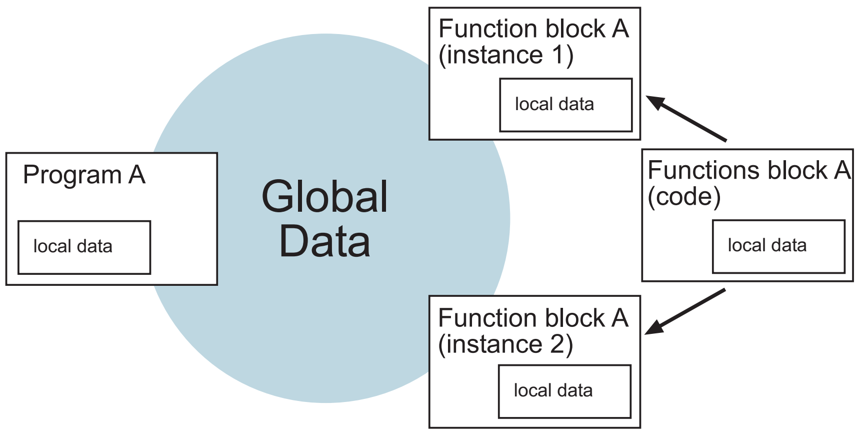


Fig. 4-5: Instancing

For each instance, a copy of the data areas is created.

Function block A is instantiated in two calls: For each call there is a structure (with the name of the instance) which contains the call-specific variable values without side effects.

Instance 1 and instance 2 in the above example are local data (structures) in the calling program. These local instance data can be input variables for other function blocks or programs.
This instancing, which is rather unusual for the traditional PLC programmer, enables a side effect-free processing of variables and their diagnosis with the programming tool, which has been standard in modern PC-based programming environments. The convenient benefit of the absolute lack of side effects for the diagnosis of the program flow by far justifies the additional complication of the instancing: In older PLC environments, the code of an FB was copied several times and program changes were then done in each code copy.

Programs

Programs are superior program organization units: A program calls up functions and function blocks, in some implementations also other programs. They can be written in all languages.
In contrast to function blocks, programs are not instanced. They do not feature a memory function for local data if they are called up more than once.

Program example:

```
PROGRAM Main
(* Declaration part*)
```

```
VAR
  counter_1 : Counter; (* instance of FB counter *)
  iActCount : INT;
END_VAR
(* Instruction part*)
IF bfirstCycle THEN
  counter_1(Mode := 0);
  (* -> Call of FB counter with reset mode *)
ELSE
  counter_1(Mode := 1);
  (* -> Call of FB counter with counter mode*)
END_IF
actCount := counter_1.Out;
  (* -> access to output variable of counter_1*)
END_PROGRAM
```

4.2.4 The Programming Languages

Sequential Function Chart (SFC), Instruction List (IL), Ladder Diagram (LD), Function Block Diagram (FBD) and Structured Text (ST) are supported as programming languages. Each language is suited for special applications and for the solution of particular problems. The syntax of the programming languages is exactly specified in the norm, so that the user can be sure that at least the identical syntax is used in all IEC 61131 compatible development packages.

Instruction List (IL)

Instruction List (IL) is the assembler among the programming languages for PLCs. Regarded as the mother of PLC programming, IL is mainly used in Europe. IL is excellent for processing simple sequential programmes. If loop constructions are necessary, however, IL has a tendency to get rather unclear.

IL is a low-level language derived from earlier assemblers, working accumulator-oriented. Per line of program code, just one action (such as load in accumulator, save in register) can be performed. Flow control, such as branches, are executed with coordinated and uncoordinated jumps and jump marks. Comments are added after the control statements in the line.

Each instruction begins in a new line and comprises an operator and, depending on the kind of operation, one or several operands, separated by commas.
In front of each instruction, an identification mark can be used, followed by a colon (:) .
A comment has to be the last element in a line. Empty lines can be inserted between instructions.

Example:

Jump mark	Operator	Operand	Comment
	LD	17	
	ST	lint	(* comment *)
	GE	5	
	JMPC	next	
	LD	idword	
	EQ	instruct.sdword	
	STN	test	
next:			

Modifiers and Operators in IL

In IL, the following modifiers and operators can be used.

Modifiers:

- C for JMP, CAL, RET: The instruction is only executed if the result of the preceding expression is TRUE.
- N for JMPC, CALC, RETC: The instruction is only executed if the result of the preceding expression is FALSE.
- N for others: Negation of the operand (not of the accumulator)

In the following table all operators in IL are listed with their possible modifiers and their corresponding meaning.

Operator	Modi-fiers	Meaning
LD	N	set actual result equal to operand
ST	N	save actual result in place of operand
S		set Bool operand to TRUE exactly if actual result is TRUE
R		set Bool operand to FALSE exactly if actual result is TRUE
AND	N,(	bitwise AND
OR	N,(	bitwise OR
XOR	N,(	bitwise exclusive OR
ADD	(	addition
SUB	(	subtraction
MUL	(	multiplication
DIV	(	division
GT	(	>
GE	(	>=
EQ	(	=
NE	(	<>
LE	(	<=
LT	(	<
JMP	CN	jump to mark
CAL	CN	call up function block
RET	CN	return from calling function block
)		evaluate reset operation

Table 4-10: Operators in IL

See also „Operators“, page 71.

Example for an IL program using some modifiers:

Jump mark	Operator	Operand	Comment
	LD	TRUE	(* load TRUE into accu-mulator *)
	ANDN	BOOL1	(* execute AND with negated value of variable BOOL1 *)
	JMPC	mark	(* if result was TRUE, jump to mark „mark“ *)
	LDN	BOOL2	(* save negated value of *)
	ST	ERG	(* BOOL2 in ERG *)
marke:	LD	BOOL2	(* save value of *)
	ST	ERG	(* BOOL2 in ERG *)

In IL, brackets can also be used after an operation. The value in the brackets is then taken as operand.

For example:

```
LD   2
MUL  2
ADD  3
ST   Erg
```

Here the value of Erg is 7. However, if brackets are used:

```
LD   2
MUL( 2
ADD  3
)    ST   Erg
```

Here the value for Erg is 10, as the operation MUL will not be evaluated until „)“ is reached; consequently, 5 is calculcated as operand for MUL.

Structured Text (ST)

The programming language ST (Structured Text) is a high-level language that resembles C and Pascal. The language syntax resembles that of Pascal. Under ST, efficient loop programming is possible without branch instructions. In addition, mathematical functions can be mapped in an excellent way: Iterations (For, While, Repeat) and conditional instructions (If ... Then ... Else, Case) are part of this language. Practical experience has shown that ST constructions are well readable and comprehensible - similar to Pascal. PLC programmers in Europe often choose ST as the mainly used language if they can use the language variety of IEC 1131.

Structured Text consists of a number of instructions, which can be executed either conditional, as in high-level languages, („IF..THEN..ELSE“) or in loops (WHILE..DO).

Instructions always end with ";".

Example:

```
IF value < 7 THEN
  WHILE value < 8 DO
    value := value + 1;
  END_WHILE;
END_IF;
```

<u>Expressions</u>

An expression is a construct which delivers a value after being evaluated.

Expressions are comprised of operators and operands. An operand can be a constant expression, a variable, a function call or another expression.

Evaluation of expressions

An expression is evaluated by processing the operators according to certain binding rules. The operator featuring the strongest binding is processed first, the operator with the second strongest binding next etc. until all operators are processed.

Operators of equal binding strength are processed from left to right.

PD_UserMan_IEC_us.fm

In the following table the ST operators are shown according to their binding strength.

Operation	Symbol	Binding strength
bracketing	(<expression>)	strongest
function call	<function name> (<parameter list>)	
exponentiation	**	
negation	-	
complement formation	NOT	
multiplication	*	
division	/	
modulo	MOD	
addition	+	
subtraction	-	
comparison	<,>,<=, >=	
equality	=	
inequality	<>	
Bool AND	AND	
Bool XOR	XOR	
Bool OR	OR	weakest

Table 4-11: Binding strength of operations

The table below shows the instructions available in ST, along with examples.

Instruction type	Example
assignment	A:=B; CV := CV + 1; C:=SIN(X);

call of an FB and use of the FB output	CMD_TMR(IN := %IX5, PT := 300);
	A:=CMD_TMR.Q

RETURN	RETURN;

IF	D:=B*B;
	IF D<0.0 THEN
	  C:=A;
	ELSIF D=0.0 THEN
	  C:=B;
	ELSE
	  C:=D;
	END_IF;

Instruction type	Example

CASE	CASE INT1 OF
	1:BOOL1 := TRUE;
	2:BOOL2 := TRUE;
	ELSE
	BOOL1 := FALSE;
	BOOL2 := FALSE;
	END_CASE;

FOR	J:=101;
	FOR I:=1 TO 100 BY 2 DO
	  IF ARR[I] = 70 THEN
	    J:=I;
	    EXIT;
	  END_IF;
	END_FOR;

WHILE	J:=1;
	WHILE J<= 100 AND ARR[J] <> 70 DO
	  J:=J+2;
	END_WHILE;

REPEAT	J:=-1;
	REPEAT
	  J:=J+2;
	UNTIL J= 101 OR ARR[J] = 70
	END_REPEAT;

EXIT	EXIT;
empty command	;

Table 4-12: Commands in ST

<u>Instructions in Structured Text</u>

As the name suggests, Structured Text was developed for structu-red programming. It offers set structures for programming certain constructs that are used rather often, such as loops.

The advantages are a lower error probability and a more structured program.

Here is a comparison of equal program sequences in IL and ST:

A loop for calculating powers of two in IL:

Loop:

```
LD   Counter
EQ   0
JMPC end

LD   Var1
MUL  2
ST   Var1
```

```
LD   Counter
SUB  1
ST   Counter
JMP  Loop

end:
LD   Var1
ST   Erg
```

The same loop programmed in ST would be:

```
WHILE Counter<>0 DO
  Var1:=Var1*2;
  Counter:=Counter-1;
END_WHILE

Erg:=Var1;
```

As can be seen, in ST, the loop can not only be programmed shorter, it is also much easier to read, in particular when imagining loops that are nested into one another in larger constructions.

The various structures are described below.

<u>Assignment operator</u>

On the left side of an assignment is an operand (variable, address), to which the value of the expression on the right side is assigned with the assignment operator :=

Example:

```
Var1 := Var2 * 10;
```

After this line is executed, Var1 has the tenfold value of Var2.

<u>Call of function blocks in ST</u>

A function block in ST is called by writing the name of the instance of the function block and assigning the desired parameter values in brackets. In the following example, a timer is called with assignments for the parameters IN and PT. In a next step, the derived variable Q is assigned to variable A.

As in IL, the calculated variable is addressed with the name of the function block, followed by a dot and the name of the variable:

```
CMD_TMR(IN := %IX5, PT := 300);
A:=CMD_TMR.Q
```

<u>RETURN instruction</u>

The RETURN instruction can be used to end a function, e.g. dependent on a condition.

Example:

```
IF COUNTER > 10 THEN
  RETURN;
END_IF;
```

<u>IF instruction/alternative branching</u>

With the IF instruction, a condition can be checked and dependent on this condition, instructions can be executed.

Syntax:

```
IF <Boolean_expression1> THEN
  <IF_instructions>
{ELSIF <Boolean_expression2> THEN
  <ELSIF_instructions1>
.
.
ELSIF <Boolean_expression n> THEN
  <ELSIF_instructions n-1>
ELSE
  <ELSE_instructions>}
END_IF;
```

The part in curly brackets {} is optional.

If <Boolean_expression1> is TRUE, then only the <IF_instructions> will be executed. None of the other instructions will be executed.

Otherwise, the Boolean expressions will be evaluated in line beginning with <Boolean_expression2> until one of the expressions is TRUE. Then, only the instructions after this Bool expression and before the next ELSE or ELSIF will be evaluated.

If none of the Boolean expressions is TRUE, then only the <ELSE_instructions> will be evaluated.

Example:

```
IF temp<17 THEN
  heating_on := TRUE;
ELSE
  heating_on := FALSE;
END_IF;
```

In this case, the heating will be switched on if the temperature falls below 17 degrees centigrade, otherwise, it will remain off.

<u>CASE instruction/multiple choice</u>

With the CASE instruction, several conditional instructions with the same conditional variable can be combined in one construct.

Syntax:

```
CASE <Var1> OF
<value 1>:<instruction 1>
<value 2>:<instruction 2>
     ...
<Wert n>:<instruction n>
ELSE<ELSE-instruction>
END_CASE;
```

A CASE instruction is processed according to the following scheme:

- If the variable in <VAR1> has the value <value i>, the instruction <instruction i> will be executed.
- If <Var 1> has none of the given values, the <ELSE instruction> will be executed.
- If the same instruction has to be executed for several values of the variable, these values can be written one after another, separated by a comma. Thereby, the joint instruction can be effected.

Example:

```
CASE INT1 OF
1, 5:BOOL1 := TRUE;
  BOOL3 := FALSE;
2: BOOL2 := FALSE;
  BOOL3 := TRUE;
ELSE
  BOOL1 := NOT BOOL1;
  BOOL2 := BOOL1 OR BOOL2;
END_CASE;
```

<u>FOR loop/counting loop</u>

With the FOR loop, repeated procedures can be programmed.

Syntax:

```
INT_Var :INT;
FOR <INT_Var> := <INIT_VALUE> TO <END_VALUE> {BY <step
  size>} DO
  <instructions>
END_FOR;
```

The part in curly brackets {} is optional.

The <instructions> will be carried out as long as the counter <INT_Var> is not bigger than <END_VALUE>. This is checked before the <instructions> are carried out, so that the <instructions> are never carried out if <INIT_VALUE> is greater than <END_VALUE>.

Always when <instructions> was executed, <INT_Var> is increased by <stepsize>. The step size can have any integer value. If it is missing, it is set to 1. Since <INT_Var> can only get bigger, the loop must terminate.

Example:

```
FOR counter:=1 TO 5 BY 1 DO
Var1:=Var1*2;
END_FOR;
Erg:=Var1;
```

Assuming that variable Var1 was pre-assigned with the value 1, then it will have the value 32 after the FOR loop.

WHILE loop/rejecting loop

The WHILE loop can be used like the FOR loop, with the exception that the truncation condition can be any Boolean expression. That means you set a condition which, if it is true, will result in the execution of the loop.

Syntax:

```
WHILE <Boolean expression> DO
  <instructions>
END_WHILE;
```

The <instructions> are repeated as long as <Boolean expression> is TRUE. If <Boolean expression> is FALSE in the first evaluation, then the <instructions> will never be carried out. If <Boolean expression> is never FALSE, then the <instructions> will be repeated endlessly, which will cause a run time error.

NOTE

The programmer has to make sure that there is no endless loop by changing the condition in the instruction part of the loop, e.g. by counting upward or downward a counter. If there is an endless loop, the the diagnosis *Cycle time transgression* will be triggered in the PacController.

Example:

```
WHILE counter<>0 DO
  Var1 := Var1*2;
  counter := counter-1;
END_WHILE
```

In a way, the WHILE loop and the REPEAT loop are both more powerful than the FOR loop, since the number of loop runs does not have to be known before the loop is executed. In some cases, these two loop types will be sufficient. However, if the number of loop runs is clear, a FOR loop should be chosen, since it will not allow an endless loop.

REPEAT loop/non-rejecting loop

In the REPEAT loop, the truncation condition will be checked only after the loop is executed. In consequence, the loop is run at least once irrespective of the truncation condition.

Syntax:

```
REPEAT
  <instructions>
UNTIL <Boolean expression>
END_REPEAT;
```

The <instructions> are carried out as long as <Boolean expression> is TRUE.

If <Boolean expression> is TRUE in the first evaluation, then the <instructions> will be carried out just once. If <Boolean expression> is never TRUE, then the <instructions> will be repeated endlessly, which will cause a run time error.

NOTE

The programmer has to make sure that there is no endless loop by changing the condition in the instruction part of the loop, e.g. by counting upward or downward a counter. If there is an endless loop, the diagnosis *Cycle time transgression* will be triggered in the PacController.

Example:

```
REPEAT
  Var1 := Var1*2;
  counter := counter-1;
UNTIL
  counter=0
END_REPEAT
```

EXIT instruction

If FOR, WHILE or REPEAT loops contain the EXIT instruction, the innermost loop will be completed, independent of the truncation condition.

Sequential Function Chart (SFC)

The programming language Sequential Function Chart (SFC) is a graphic language to represent a state machine, which in one cycle carries out a single transition to a (next) action state. The graphic representation of transitions and actions resembles a flow diagram, is easy to read and well suited for programming superior state sequences. Programmers often fall victim to the misunderstanding that the program path is completely processed in the program

cycle. This is not the case. For that reason, fast complex calculations cannot be executed well in SFC but only in the reserved action modules, which in turn can be filled in any language.
Under TRUE, the transitions between action states become active as Boolean equations. SFC allows both alternative and parallel branches:

- Actions in alternative branches are executed if the respective entry transition is fulfilled. The check sequence (priority) is set from left to right.
- Actions in simultaneous branches are all started together. A transition at the end of the simultaneous branch defines the exit event.

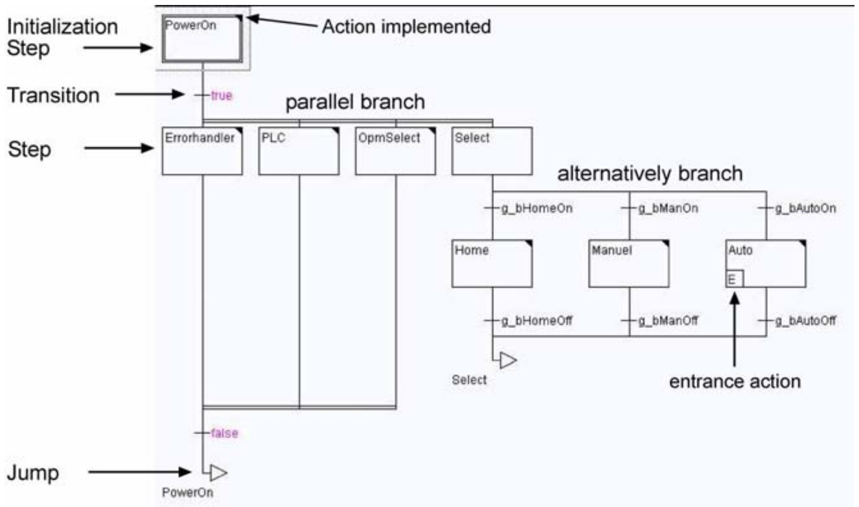


Fig. 4-6: Example for a network in SFC

<u>Step</u>

A block that is written in Instruction List is made up of a sequence of steps, which are linked with one another via directed connections (transitions).

Two kinds of steps can be distinguished.

- The simplified form consists of an action and a marker, which indicates whether the step is active. If an action is implemented to a step, a small triangle appears in the upper right corner of the step.

- An IEC step consists of a marker and one or more assigned actions. The associated actions appear to the right of the step. More details will be given later.

Action

An action can contain a sequence of instructions in IL or ST, a quantity of networks in FBD or LD or again a sequential structure.

In simplified steps, an action is always linked to a step. In order to edit an action, doubleclick on the step to which the action belongs or highlight the step and choose the menu item EXTRAS | ZOOM ACTION / TRANSITION.

Actions of IEC steps are arranged in the Object Organizer directly under their SFC module. They are loaded into their editor by dou-bleclicking or pressing <ENTER>. New actions can be created with PROJECT | ADD ACTION.

Input and Output Actions

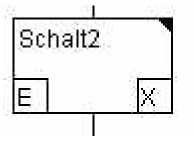
An input and an output action can be added to a step. An input action is executed just once, immediately after the step is activated. An output action is executed just once before the step is deactivated.

A step with an input action is marked with an 'I' in the lower left cor-ner while the output action is marked with an 'X' in the lower right corner.

The input and output actions can be implemented in any language. In order to edit input and output actions, doubleclick on the respec-tive corner in the step.

Input and output actions can only be defined as a simplified step, not to an IEC step.

Example of a step with input and output actions:



Transition / Transition Conditions

Inbetween steps, there are so-called transitions.

A transition condition can be a Boolean variable, an address, a con-stant expression or a sequence of instruction with a Boolean result in any language.

Active step

After the SFC module is called, the action belonging to the initial step (double edged) will be executed. A step the action of which is being executed is called active. If the step is active, the respective action is executed once per cycle. In online mode, active steps are blue.

To each step belongs a flag which stores the state of the step. The step marker (active or inactive state of the step) is represented by the logic value of a Boolean structural element <StepName>.x. This Boolean variable is TRUE if the step is active and FALSE if the step is inactive. This variable is declared implicitly and can be used in each action and transition of the SFC block.

In a control cycle, all actions which belong to active steps are executed. Next, the respective following steps of active steps will become active if the transition conditions of the subsequent steps are TRUE. The now active steps will only be executed in the next cycle.

IEC step

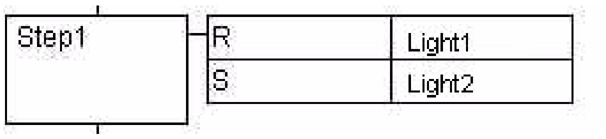
In addition to simplified steps, norm-conformous IEC steps are available in SFC.

Any number of actions can be assigned to an IEC step. The actions of IEC steps are available separately from the steps and can be used more than once within their block. For that purpose, they have to be associated with the individual steps with the command EXTRAS | ASSOCIATE ACTION.

Alongside actions, also Boolean variables can be assigned to steps. The actions and Boolean variables can be activated and deactivated with so-called qualifiers, partly with time delays. Since an action can still be active even if the next step is already being processed, e.g. by qualifier S (Set), ancillarities can be achieved.

The associated actions to an IEC step are indicated to the right of the step in a split box. The left field contains the qualifier and maybe also a time constant while the right field contains the name of the action.

Example for an IEC step with two actions:



In order to make it easier to follow the procedures, all actions in online mode are blue, like the active steps. After each cycle it is checked which actions are active.

Whether a newly inserted step is an IEC step depends on whether the menu command EXTRAS USE IEC STEPS is chosen.

In the Object Organizer, the actions are arranged directly under their IL block. New actions can be created with PROJECT | ADD ACTION.

NOTE

In order to be able to use IEC steps, the project has to be embedded in the special SFC library *Iecsfc.lib*.

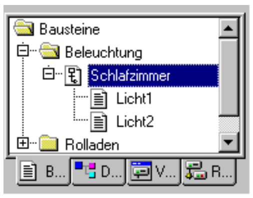


Fig. 4-7: SFC block with actions in object organizer

Qualifiers

The following qualifiers are available to associate actions to IEC steps:

Qualifier	Meaning	Remark
N	Non-stored	action is active as long as the step
R	overriding Reset	action is deactivated
S	Set (Stored)	action is activated and remains active until reset
L	time Limited	action is activated for a certain time
D	time Delayed	action becomes active after a certain time, if the step is still active
P	Pulse	action is executed exactly once if the step is active
SD	Stored and time Delayed	action is activated after a certain time and remains active until reset
DS	Delayed and Stored	action is activated after a certain time, if the step is still active, and remains active until reset
SL	Stored and time Limited	action is activated for a certain time

Table 4-13: Qualifier in SFC

Alternative branch

Two or more branches in SFC can be defined as alternative branches. Each alternative branch has to begin and end with a transition. Alternative branches can contain simultaneous branches and further alternative branches. An alternative branch begins at a horizontal line (alternative start) and ends at a horizontal line (alternative end) or with a branch.

If the step preceding the alternative start line is active, the first transition of each alternative branch will be evaluated from left to right. The first transition from the left that carries a TRUE transition condition will be opened and the following steps will become active (see active step).

Parallel branch

Two or more branches in SFC can be defined as parallel branches. Each parallel branch has to start and to end with a step. Parallel branches can contain alternative branches or further parallel branches. A parallel branch begins at a double line (simultaneous beginning) and ends at a double line (parallel start) or at a branch.

If the step preceding the parallel start line is active and the transition condition after this step is TRUE, the first steps of all parallel branches become active (see active step). The branches are then processed simultaneously to one another. The step after the parallel end line becomes active if all earlier steps are active and the transition condition preceding the step is TRUE.

Jump

A jump is a connection to the step the name of which is indicated under the jump symbol. Jumps are needed since it is not allowed to create ascending or crossing connections.

The diagram shows a simple procedure of actions in stored, normal and reset operating mode:

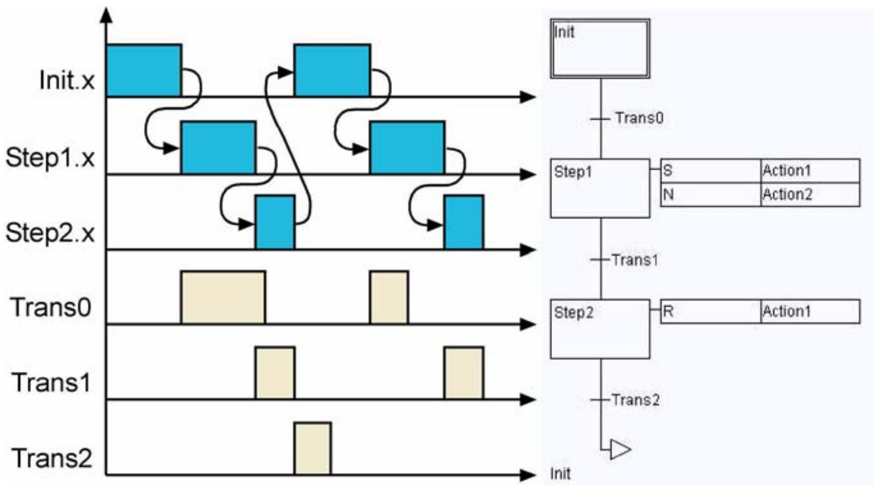


Fig. 4-8: The transitions a, b and c control the development of steps 1 and 2.

The action qualifiers control the execution of the code independent of the active step: Stored actions will not become inactive before the following steps.

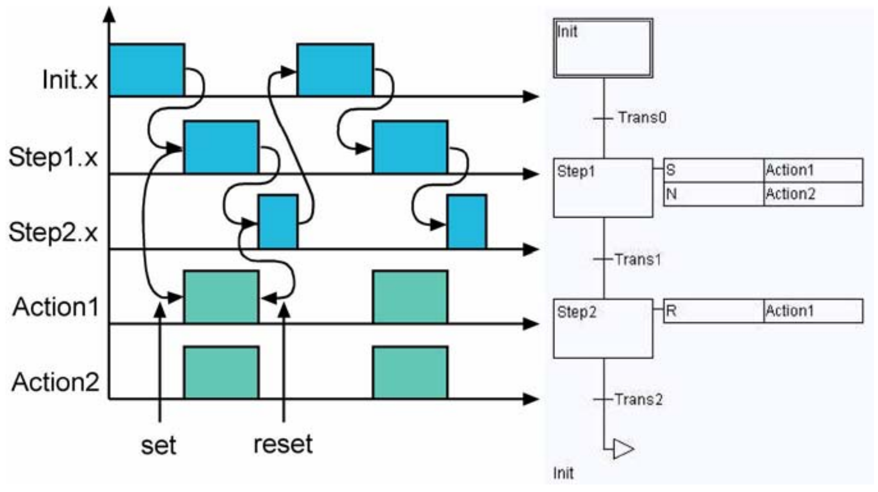


Fig. 4-9: Action 1 is terminated only by Reset in step 2, not by end of step 1.

The action qualifiers L and D in their operating mode: limited processing by L and delayed start of the processing until the end of the step by D.

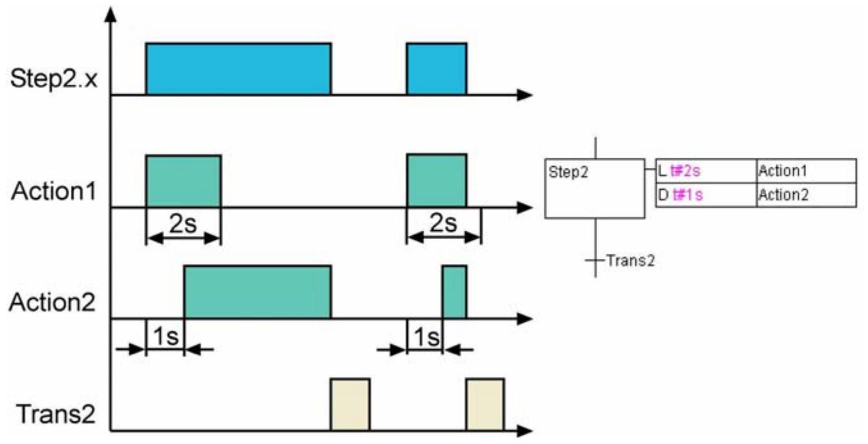


Fig. 4-10: Action qualifyers L and D for time control of actions.

Function Block Diagram (FBD), Function Block Language (FBL)

The graphic programming language Function Block Diagram can be used to realize even complicated control tasks. Based on defined function blocks, any program sequences can be realized with the help of connecting elements. Data flow through the program can be presented in a pattern, thus helping make the program sequences transparent. A loop construct or branches can be difficult. Often, also hardware components are offered with the respective function blocks, so that corresponding modules are available on both hardware and software levels.

Outputs of function blocks are linked with inputs of the following blocks. Data flow can be presented in full graphic. Jump and return jump facilitate programming.

Function Block Diagram is a programming language with a graphic orientation. It uses a list of networks, with each network containing a structure representing a logic/arithmetic expression, the call of a function block, a branch or a return instruction.

Example for a network in Function Block Diagram, as it could typically look like in EPAS-4:

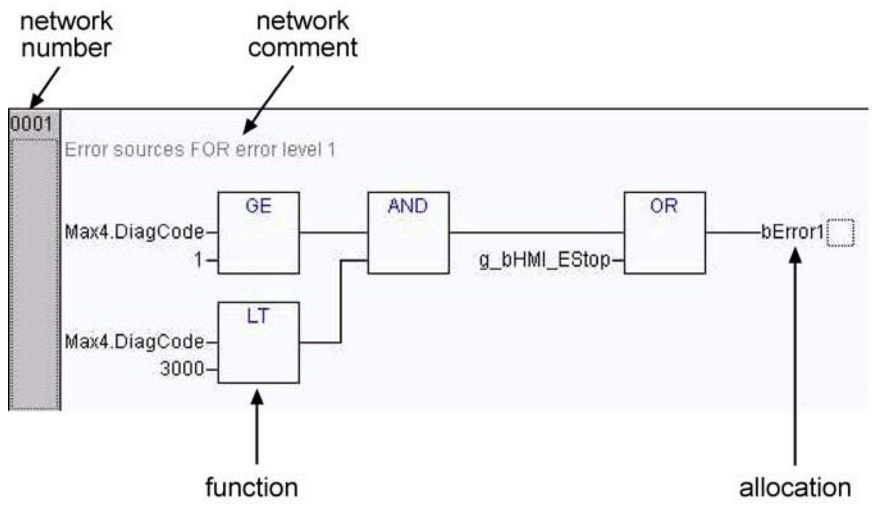


Fig. 4-11: Network in function block diagram

Ladder Diagram (LD)

A program in Ladder Diagram (LD) is regarded as the easiest way to give controls well-defined functionality. LD is particularly advantageous if the control is to be implemented as a replacement for a wired logic or if basic functions, such as start-up locks, are to be programmed. LD is particularly well suited for complex AND / OR logics. Since it is so easy, it is the prefered language e.g. in the US. As the norm provides for the start of complex blocks (written in other languages) with the help of an enable input, powerful constructions can be controlled with Ladder logic.
The graphic representation is the simulation of a current flow through a „left contact rail“ through input switch (represented by variables) via current paths to output actuators (coils), which are also represented through variables. Branches and returns to the beginning are allowed.

The Ladder Diagram is also a graphic-oriented language. In principle, the Ladder Diagram is similar to the principle of an electric circuit/.

On the one hand, the Ladder Diagram is good for constructing sequential logic systems. On the other hand, it can also be applied to create networks like in FBD. Thus, Ladder Diagram is also excellent to control the call of other modules. More details are given in a later chapter.

The Ladder Diagram consists of a sequence of networks. A network is restricted on the left and right sides by a left and a right vertical power line. Inbetween the power lines, there is a wiring diagram of contacts, coils and connecting lines.

On the left side, each network consists of a sequence of contacts which transmit the states ON or OFF from left to right. These states correspond to the Boolean values TRUE and FALSE. To each contact belongs a Boolean variable. If the variable is TRUE, the state will be transmitted via the connecting line from left to right. Otherwise, the right connection has the value OFF.

Example for a network in the Ladder Diagram, as it could typically look in EPAS-4:

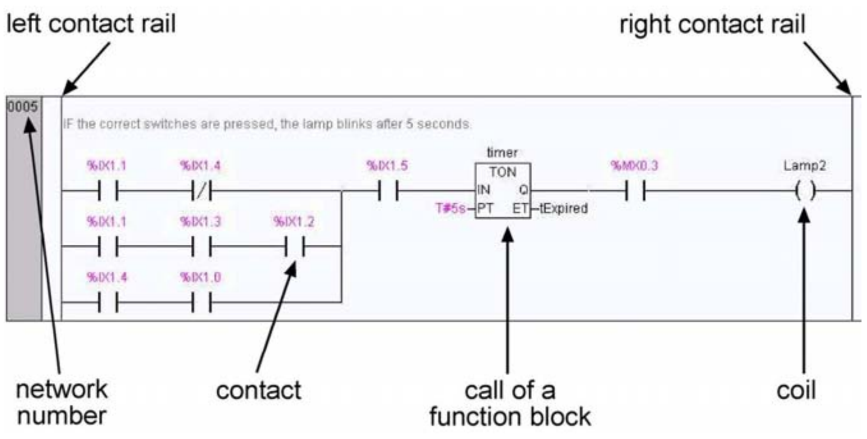


Fig. 4-12: Network in Ladder Diagram

<u>Contact </u>

On the left side, each network in the Ladder Diagram consists of a network of contacts (represented by two parallel lines: | |), which transmit the state ON or OFF from left to right.

These states correspond to the Boolean values TRUE and FALSE. To each contact belongs a Boolean variable. If the variable is TRUE, the state will be transmitted via the connecting line from left to right. Otherwise, the right connection has the value OFF".

Contacts can be switched parallel. In this case, one of the parallel branches has to transmit the value ON for the parallel branch to transmit the value ON. If the contacts are switched in line, all contacts have to transmit the state „On" for the last contact to transmit the state ON. The contact connections correspond to an electric parallel connection and serial connection, respectively.

A contact can also be negated. This is indicated by a slash in the contact symbol: |/|. In this case, the value of the line is transmitted if the variable is FALSE.

Coil

On the left side of a network in the Ladder Diagram any number of so-called coils can be found. Coils are represented by brackets: (). Coils can only be switched parallel. A coil transmits the value of the connection from left to right and copies it into a corresponding Boolean variable. The values On and Off can be set on the input line. On corresponds to the Boolean variable TRUE and Off to the Boolean variable FALSE.

Contacts and coils can also be negated. (In the example, contact SWITCH1 and coil %QX3.0 are negated.) If a coil is negated (indicated by a slash in the coil symbol: (/)), it copies the negated value to the corresponding Boolean variable. If a contact is negated, it interconnects only if the Boolean variable is FALSE.

Function blocks in Ladder Diagram

In addition to contacts and coils, also function blocks and programs can be entered. In the network, they have to have an input and an output with Boolean values. They can be used at the same sites as contacts, i.e., on the left side of the Ladder Diagram network.

Set/reset coils

Coils can also be defined as Set or Reset coils. A set coil (represented by an ’S’ in the coil symbol (S)) never overwrites the value TRUE in the respective Boolean variable. That is, if a variable was once set on TRUE it will also remain so.

A Reset coil (’R’ in the coil symbol (R)) never overwrites the value FALSE in the respective Boolean variable. If the variable was once set to FALSE, it will remain so.

LD as FBD

When working with Ladder Diagram, you might want to use the result of the contact connection to control other blocks. Then you can file the result with the help of the coils in a global variable that is used further elsewhere. You can also fit the possible call directly into your Ladder Diagram network. In order to do so, insert a module with and EN input.

Such blocks are conventional operands, functions, programs or function blocks that feature an additional input marked with EN. The EN input is always of the Boolean type. It has the following meaning: The block featuring the EN input will be evaluated if EN is TRUE.

An EN block is switched parallel to the coils, while the EN input is connected with the connecting line between the contacts and coils. If the information ON is transmitted via this line, the block will be evaluated in the usual way.

Taking such an EN block as the basis, networks can be created like in FBD.

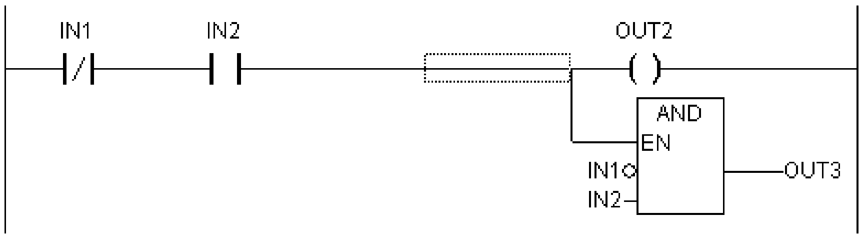


Fig. 4-13: Part of an LD network with an EN block

4.2.5 Operators

In contrast to standard functions, IEC operators are implicitly known in the complete project. In the block implementation, operators are used like functions.

The list blow shows all operators that are supported:

Overview

In expressions, operands are linked by operators.

Operator / Symbol	Short description	Explanation
[]	start and end for : 1. field index (access to field element) 2. string length (for declaration)	
(...)	bracketing	A:= (5*2) + (4/2) -> A = 12
function name (argument list)	function evaluation	ADD (A, B, C)
EXPT **	exponentiation (** currently not supported)	A:= 2**3 -> A = 8
-	negation	A:= -5 -> A = -5
NOT	complement	
MUL *	multiplication	A:= 5*2 -> A = 10
DIV /	division	A:= 4/2 -> A = 2
MOD	modulo (not supported for REAL and LREAL.)	A:= 12 MOD 10 -> A = 2
ADD +	addition	A:= 3+4 -> A = 7
SUB -	subtraction	A:= 4-2 -> A = 2
>	comparison operator greater than	
<	comparison operator less than	
>=	comparison operator equal to or greater than	
<=	comparison operator equal to or less than	

Operator / Symbol	Short description	Explanation
=	comparison operator equal to	
<>	comparison operator not equal to	
AND &	boolean AND	
XOR	boolean exclusive OR	
OR	boolean OR	
MOVE :=	allocation 1. operator for start value allocation 2. input connection operator (allocation of actual parameters to formal parameters at POU call) 3. instruction operator	

Table 4-14: Operators

Operators of equal rank are evaluated from left to right..

Rank	Operation	Operator
highest		[]
	bracketing	(...)
	function evaluation	function name (argument list)
	exponentiation	**
	negation	-
	complement	NOT
	multiplication	*
	division	/
	modulo	MOD
	addition	+
	subtraction	-
	comparison	>, <, >=, <=
	equality	=
	inequality	<>
	AND	AND, &
	exclusive OR	XOR
	OR	OR

Rank	Operation	Operator
lowest	allocation	:=

Table 4-15: Operators - priority and associativity

Further operators:

Name	Function	Description
*_TO_**		
TRUNC	conversion of a REAL into an INT value	
BCD_TO_**	conversion of a BCD value e.g. into an INT	
*_TO_BCD	conversion e.g. of an INT value into a BCD value	

Table 4-16: Operators for type conversion

*	Data type of the input, e.g. REAL
**	Data type of the output, e.g. INT

Further operators:

Name	Function	Description
ABS	absolute value	F:= | IN |
SQRT	square root	F:= √IN
LN	natural logarithm	F:= $\log_e$ (IN)
LOG	logarithm to the base of 10	F:= $\log_{10}$ (IN)
EXP	exponent to the base e	F:= eIN
SIN	sinus, IN in radial measure	F:= SIN (IN)
COS	cosine, IN in radial measure	F:= COS (IN)
TAN	tangent, IN in radial measure	F:= TAN (IN)
ASIN	Arcsin, main value	F:= ARCSIN (IN)
ACOS	Arccos, main value	F:= ARCCOS (IN)
ATAN	Arctan, main value	F:= ARCTAN (IN)
EXPT	exponentiation of one variable with another	F:= IN1IN2

Table 4-17: Numeric operators

PD_UserMan_IEC_us.fm

Arithmetic Operators

ADD

Addition of variables of the BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL and LREAL types.

Also two TIME variables can be added. Their sum is again a time (e.g. t#45s + t#50s = t#1m35s)

Example in IL:

```
LD   7
ADD  2,4,7
ST   var1
```

Example in ST:

```
var1 := 7+2+4+7;
```

MUL

Multiplication of variables of the BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL and LREAL types.

Example in IL:

```
LD   7
MUL  2,4,7
ST   var1
```

Example in ST:

```
var1 := 7*2*4*7;
```

SUB

Subtraction of a variable of the BYTE, WORD, DWORD, SINT, USINT, INT, DINT, UDINT, REAL and LREAL types from another variable of one of the above types.

A TIME variable can also be subtracted from another TIME variable, the result is again a TIME type. Note that negative TIME values are not defined.

Example in IL:

```
LD   7
SUB  8
ST   var1
```

Example in ST:

```
var1 := 7-2;
```

DIV

Division of a variable of the BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL und LREAL types by another variable of one of the same types.

Example in IL:

```
LD 8
DIV2
STvar1
```

Example in ST:

```
var1 := 8/2;
```

<u>MOD</u>

Modulo Division of a variable of the BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL and LREAL types by another variable of one of the same types. The function delivers as result the integer remainder of the division.

Example in IL:

```
LD   9
MOD  2
ST   var1 (* result is 1 *)
```

Example in ST:

```
var1 := 9 MOD 2;
```

<u>INDEXOF</u>

This function delivers as result the internal index of a block.

Example in ST:

```
var1 := INDEXOF(baustein2);
```

<u>SIZEOF</u>

This function delivers as result the number of bytes required by the data type.

Example in IL:

```
arr1:ARRAY[0..4] OF INT;
var1:=INT;
LD   arr1
SIZEOF
ST   var1    (* result is 10 *)
```

Example in ST: :

```
   var1 : INT;
pt := ADR(pt^) + SIZEOF(INT);
```

Bitstring operators

<u>AND</u>

Bitwise AND of bit operands. The operands should be of the BOOL, BYTE, WORD or DWORD types.

Example in IL:

```
var1 :BYTE;
LD   2#1001_0011
AND  2#1000_1010
ST   var1 (* result is 2#1000_0010 *)
```

Example in ST:

```
var1 := 2#1001_0011 AND 2#1000_1010
```

<u>OR</u>

Bitwise OR of bit operands. The operands should be of the BOOL, BYTE, WORD or DWORD types.

Example in IL:

```
var1 :BYTE;
LD   2#1001_0011
OR   2#1000_1010
ST   var1 (* result is 2#1001_1011 *)
```

Example in ST:

```
Var1 := 2#1001_0011 OR 2#1000_1010
```

XOR

Bitwise XOR of bit operands. The operands should be of the BOOL, BYTE, WORD or DWORD type.

Example in IL:

```
Var1 :BYTE;
LD   2#1001_0011
XOR  2#1000_1010
ST   Var1 (* result is 2#0001_1001 *)
```

Example in ST:

```
Var1 := 2#1001_0011 XOR 2#1000_1010
```

NOT

Bitwise NOT of a bit operand. The operand should be of BOOL, BYTE, WORD or DWORD type.

Example in IL:

```
Var1 :BYTE;
LD2#1001_0011
NOT
STVar1 (* result is 2#0110_1100 *)
```

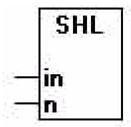
Example in ST:

```
Var1 := NOT 2#1001_0011
```

Bit-shift operators

The following operators are represented with a map as FBD operators.

SHL



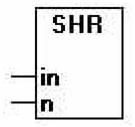
Bitwise left shift of an operand: A:= SHL (IN, N)

A, IN and N should be of the BYTE, WORD, DWORD type. IN is shifted to the left by N bits and filled up with zeros from the right.

Example:

```
LD   1
SHL  1
ST   Var1 (* result is 2 *)
```

SHR



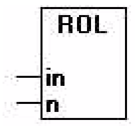
Bitwise right shift of an operand: A:= SHR (IN, N)

A, IN and N should by of the BYTE, WORD or DWORD type. IN is shifted to the right by N bits and filled up with zeros from the left.

Example:

```
LD   32
SHL  2
ST   Var1(* result is 8 *)
```

ROL



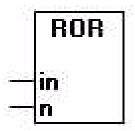
Bitwise left rotation of an operand: A:= ROL (IN, N)

A, IN and N should be of the BYTE, WORD or DWORD type. IN is shifted to the left N times one bit position and the most left bit is re-inserted from the right.

Example:

```
Var1 :BYTE;
LD2#1001_0011
ROL3
STVar1 (* result is 2#1001_1100 *)
```

ROR



Bitwise right rotation of an operand: A:= ROR (IN, N)

A, IN and N should be of the BYTE, WORD or DWORD type. IN is shifted to the right N times one bit position while the most right bit is re-inserted again from the left.

Example:

```
Var1 :BYTE;
LD   2#1001_0011
ROR  3
ST   Var1 (* result is 2#0111_0010 *)
```

Selection Operators

All selection operators can also be executed on variables. For illustrative reasons, the following example is restricted to constant expressions as operators.

SEL

Binary selection

OUT := SEL(G, IN0, IN1) means:

OUT := IN0 if G=FALSE;

OUT := IN1 if G=TRUE.

IN0, IN1 and OUT can be of any type, G has to be of the BOOL type. The result of the selection is IN0 if G is FALSE and IN1 if G is TRUE.

Example in IL:

```
LD   TRUE
SEL  3,4
ST   Var1 (* result is 4 *)
LD   FALSE
SEL  3,4
ST   Var1 (* result is 3 *)
```

MAX

Maximum function. Delivers the greater one of two values.

OUT := MAX(IN0, IN1)

IN0, IN1 and OUT can be of any type.

Example in IL:

```
LD   90
MAX  30
MAX  40
MAX  77
ST   Var1 (* result is 90 *)
```

MIN

Minimum function. Delivers the lesser one of two values.

OUT := MIN(IN0, IN1)

IN0, IN1 and OUT can be of any type.

Example in IL:

```
LD   90
MIN  30
MIN  40
MIN  77
ST   Var1 (* result is 30 *)
```

LIMIT

Limitation

OUT := LIMIT(Min, IN, Max) means:

OUT := MIN (MAX (IN, Min), Max)

Max is the upper, Min the lower limit for the result. If the IN value is greater than the upper limit, LIMIT will deliver Max. If IN is less than Min, the result will be Min.

IN and OUT can be of any type.

Example in IL:

```
LD      90
LIMIT   30,80
ST      Var1 (* result is 80 *)
```

MUX

Multiplexer

OUT := MUX(K, IN0,...,INn) means:

OUT := INK.

IN0, ...,INn and OUT can be of any type. K has to be of the BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT or UDINT type. MUX choses the Kth from a quantity of values.

Example in IL:

```
LD   0
MUX  30,40,50,60,70,80
ST   Var1 (* result is 30 *)
```

Comparative operators

GT

Greater than

A Boolean operator with the result TRUE, if the first operand is greater than the second operand. The operands can be of the BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME and STRING type.

Example in IL:

```
LD   20
GT   30
ST   Var1 (* result is FALSE *)
```

Example in ST:

```
VAR1 := 20 > 30 > 40 > 50 > 60 > 70;
```

LT

Less than

A Boolean operator with the result TRUE, if the first operand is less than the second operand. The operands can be of the BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME and STRING type.

Example in IL:

```
LD   20
LT   30
ST   Var1 (* result is TRUE *)
```

Example in ST: :

```
VAR1 := 20 < 30;
```

<u>LE</u>

Less than or equal to

A Boolean operator with the result TRUE, if the first operand is less than or equal to the second operand. The operands can be of the BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME and STRING types.

Example in IL:

```
LD   20
LE   30
ST   Var1 (* result is TRUE *)
```

Example in ST:

```
VAR1 := 20 <= 30;
```

<u>GE</u>

Greater than or equal to

A Boolean operator with the result TRUE, if the first operand is greater than or equal to the second one. The operands can be of the BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME and STRING types.

Example in IL: :

```
LD   60
GE   40
ST   Var1 (* result is TRUE *)
```

Example in ST:

```
VAR1 := 60 >= 40;
```

<u>EQ</u>

Equal to

A Boolean operator with the result TRUE, if the operands are equal. The operands can be of the BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME and STRING types.

Example in IL:

```
LD   40
EQ   40
ST   Var1 (* result is TRUE *)
```

Example in ST:

```
VAR1 := 40 = 40;
```

NE

Not equal to

A Boolean operator with the result TRUE, if the operands are not equal. The operands can be of the BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, U-INT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME and STRING types.

Example in IL:

```
LD   40
NE   40
ST   Var1 (* result is FALSE *)
```

Example in ST:

```
VAR1 := 40 <> 40;
```

Address operators

ADR

Address function

ADR delivers the address of its argument in a DWORD. This address can be sent to producer functions, where it can be treated like a pointer. Within the project, it can also be assigned to a pointer.

Example in IL:

```
LD   var1
ADR
ST   var2
man_fun1
```

Content operator

The dereferentiation of a pointer is effected via a content operator "^" after the pointer identifier.

Example in ST:

```
pt:POINTER TO INT;
var_int1:INT;
var_int2:INT;
pt := ADR(var_int1);
var_int2:=pt^;
```

Call operator

CAL

Call of a function block

In IL, CAL is used to call the instance of a function block. After the name of the instance follows in brackets the assignment of the input variable of the function block.

Example:

Call of the instance of a function block with assignment of the input variables Par1, Par2 on 0 or TRUE.

```
CAL INST(PAR1 := 0, PAR2 := TRUE)
```

Type conversion operators

It is not allowed to convert implicity from a “bigger“ type to a “smaller“ one (e.g. from INT to BYTE or from DINT to WORD). If you want to do that, you have to use special type conversion functions. In principle, you can convert from any elementary type to any other elementary type.

Syntax:

<elem.Typ1>_TO_<elem.Typ2>

BOOL_TO Conversion

Conversion from the BOOL type into another type:

In number types, the result is 1 if the operand is TRUE. The result is 0 if the operand is FALSE.

In the STRING type, the result is TRUE or FALSE.

Examples in ST:

```
i:=BOOL_TO_INT(TRUE);
(* -> result is 1 *)
str:=BOOL_TO_STRING(TRUE);
(* -> result is 'TRUE' *)
t:=BOOL_TO_TIME(TRUE);
(* -> result is T#1ms *)
tof:=BOOL_TO_TOD(TRUE);
(* -> result is TOD#00:00:00.001 *)
dat:=BOOL_TO_DATE(FALSE);
(* -> result is D#1970-01-01 *)
dandt:=BOOL_TO_DT(TRUE);
(* -> result isDT#1970-01-01-00:00:01 *)
```

TO_BOOL conversions

Conversion from one type into the BOOL type:

The result is TRUE if the operand is not 0. The result is FALSE if the operand is 0.

In the STRING type, the result is TRUE if the operand is TRUE. Otherwise, the result is FALSE.

Examples in ST:

```
b := BYTE_TO_BOOL(2#11010101);
(* -> result is TRUE *)
b := INT_TO_BOOL(0);
(* -> result is FALSE *)
b := TIME_TO_BOOL(T#5ms);
(* -> result is TRUE *)
b := STRING_TO_BOOL('TRUE');
(* -> result is TRUE *)
```

Conversions between integer number types

Conversion from an integer number type into another number type:

In the type conversion from greater to lower types, information can be lost. If the to be converted exceeds the range, the first bytes of the number will not be taken into consideration.

Example in ST:

```
si := INT_TO_SINT(4223); (* result is 127 *)
```

If the integer number 4223 (16#107f in hexadecimal representation) is stored in a SINT variable, the latter will contain the number 127 (16#7f in hexadecimal representation).

Example in IL:

```
LD   2
INT_TO_REAL
MUL  3.5
```

REAL_TO/ LREAL_TO conversions

Conversion from the REAL/LREAL types into another type:

The value is rounded up or down to an integer value and converted into the corresponding type. STRING, BOOL, REAL and LREAL types are exceptions.

In the type conversion from higher to lower types, information can be lost.

Example in ST:

```
i := REAL_TO_INT(1.5); (* result is 2 *)
j := REAL_TO_INT(1.4); (* result is 1 *)
```

Example in IL:

```
LD   2.7
REAL_TO_INT
GE   %MW8
```

TIME_TO / TIME_OF_DAY conversions

Conversion of the TIME and TIME_OF_DAY type into another type:

On an internal basis, time is stored in a DWORD in milliseconds (in the case of TIME_OF_DAY since 00:00). This value is converted.

In the type conversion from higher to lower types, information can be lost.

In the STRING type, the result is the time constant.

Examples in ST:

```
str :=TIME_TO_STRING(T#12ms);
(* -> result is 'T#12ms' *)
dw:=TIME_TO_DWORD(T#5m);
(* -> result is 300000 *)
si:=TOD_TO_SINT(TOD#00:00:00.012);
(* -> result is 12 *)
```

DATE_TO / DT_TO conversions

Conversions from the DATE and DATE_AND_TIME type into another type:

On an internal basis, the date has been stored in a DWORD in seconds since January 1st, 1970. This value is converted.

In the type conversion from a higher to a lower type, information can be lost.

In the STRING type, the result is a date constant. .

Examples in ST:

```
b :=DATE_TO_BOOL(D#1970-01-01);
(* -> result is FALSE *)
i :=DATE_TO_INT(D#1970-01-15);
(* -> result is 29952 *)
byt :=DT_TO_BYTE(DT#1970-01-15-05:05:05);
(* -> result is 129 *)
str:=DT_TO_STRING(DT#1998-02-13-14:20);
(* -> result is 'DT#1998-02-13-14:20' *)
```

STRING_TO conversion

Conversion from the STRING type into another type:

The operand ot the STRING type must have a valid value of the target type. Otherwise, the result is 0.

Examples in ST:

```
b :=STRING_TO_BOOL('TRUE');
(* -> result is TRUE *)
w :=STRING_TO_WORD('abc34');
(* -> result is 0 *)
t :=STRING_TO_TIME('T#127ms');
(* -> result is T#127ms *)
```

TRUNC

Conversion from the REAL type into the INT type. Only the integer fraction of the number is taken.

In the type conversion from higher to lower types, information can be lost.

Examples in ST:

```
i:=TRUNC(1.9); (* result is 1 *).
i:=TRUNC(-1.4); (* result is -1 *).
```

Example in IL:

```
LD   2.7
TRUNC
GE   %MW8
```

Numeric operators

ABS

Gives the absolute value of a number. ABS(-2) gives 2.

SQRT

Gives the square root of a number.

LN

Gives the natural logarithm of a number.l

LOG

Gives the logarithm to base 10 of a number. Liefert den Logarithmus zur Basis 10 einer Zahl.

EXP

Gives the exponential funtion.

SIN

Gives the sine of a number.

COS

Gives the cosine of a number.

TAN

Gives the tangent of a number.

ASIN

Gives the antisine (inverse function of sinus) of a number.

ACOS

Gives the arc cosine (inverse function of cosine) of a number.

ATAN

Gives the arc tangent (inverse function of tangent) of a number.

EXPT

Exponentiation of a variable by another variable:

```
OUT = IN1IN2.
```

OUT, IN1 and IN2 can be of the BYTE, WORD, DWORD, INT, DINT, REAL types.

Example in IL:

```
LD   7
EXPT 2
ST   var1 (* result is 49 *)
```

Example in ST:

```
var1 := EXPT (7,2);
```

4.2.6 Operands

In the PacDrive™, constant expressions, variables, addresses and also function calls can occur as operands.

Constants

Boolean constants

Boolean constants are the truth values TRUE and FALSE

TIME constants

In EPAS-4, TIME constants can be declared. They are used in particular in order to operate timers from the standard library. A TIME constant always consists of a leading t or T (i.e., time or TIME in the detailled version) and a double cross „#“.

The actual time declaration follows after the TIME constant. It can consist of days (d), hours (h), minutes (m), seconds (s) and milliseconds (ms). Note that the time data have to be put in order of their size (d before h before m before s before ms). However, not all times have to be included.

Examples for correct TIME constants in an ST assignment:

```
TIME1 := T#14ms;
TIME1 := T#100S12ms;
(*>Overflow in the highest component is allowed*)
TIME1 := t#12h34m15s;
```

The following example is not correct:

```
TIME1 := t#5m68s;
(* -> Overflow in a lower place*)
TIME1 := 15ms;
(* -> Es fehlt T#*)
TIME1 := t#4ms13d;
(* -> wrong order of time data*)
```

DATE constants

With the help of DATE constants, date indications can be made. A DATE constant is declared by a leading d, D, date or DATE followed by #. Thereafter, any date can be entered in the form year-month-day.

Examples:

```
DATE#1996-05-06
d#1972-03-29
```

TIME_OF_DAY constants

With the help of TIME_OF_DAY constants, times of day can be stored. A TIME_OF_DAY declaration starts with tod#, TOD#, TIME_OF_DAY# or time_of_day#. After that, a time of day can be entered in the following form: hour:minute:second. Seconds can be entered either as real numbers or fractions.

Examples:

```
TIME_OF_DAY#15:36:30.123
tod#00:00:00
```

DATE_AND_TIME constants

Date constants and times can also be combined in so-called DATE_AND_TIME constants. DATE_AND_TIME constants start with dt#, DT#, DATE_AND_TIME# or date_and_time#. The date is followed by a hyphen and the time of day.

Examples:

```
DATE_AND_TIME#1996-05-06-15:36:30
dt#1972-03-29-00:00:00
```

Number constants

Numbers can occur as dual numbers, octal numbers, decimal numbers and hexadecimal numbers. If an integer value is not a decimal number, then the base, followed by a double cross (#), has to be written in front of the integer constant. For hexadecimal numbers, the numeric values for numbers 10 to 15 are represented by the letters A to F.

Underscores within a number value are allowed.

Examples:

```
14          (decimal number)
2#1001_0011 (dual number)
8#67        (octal number)
16#A        (hexadecimal number)
```

The digit values can be of the BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL and LREAL types.

Implicit conversions from „greater“ to „lower“ types are not allowed. That is, a DINT variable cannot simply be converted into an INT variable. For such operations, the type conversion function of standard.lib has to be used (see chapter Type Conversion in the appendix).

REAL/LREAL constants

REAL and LREAL constants can be represented as decimal fraction or exponential notation. In this case, the American way of writing with a point is used.

Example:

```
7.4 instead of 7,4
1.64e+009 instead of t 1,64e+009
```

STRING constants

A string is any character sequence. String constants are delimited with inverted commas. Also blank spaces and umlaut characters can be used. They are treated just like all other characters.

In strings, the combination of a dollar sign ($) followed by two hexadecimal numbers is interpreted as hexidecimal representation of the eight bit character code. In addition, if they occur in a string, combinations of two characters starting with the dollar sign are interpreted as follows:

$$ dollar sign

$ 'inverted comma

$L or $lline feed

$N or $nnew line

$P or $ppage feed

$R or $rline break

$T or $ttab

Examples:

```
'w1Wüß?'
'Susi and Claus'
':-)'
```

Variables

Variables are either declared locally in the declaration part of a block or in the global variable lists.

Note that identifiers of variables must neither contain blank spa-cesn nor umlaut characters. Furthermore, they must not be declared double and they must not be identical with keywords. Case sensitivity is not recognized, i.e., VAR1, Var1 and var1 are not different variables. Underscores are significant in identifiers, e.g. A_BCD and AB_CD are interpreted as different identifiers. More than one underscore at the beginning of an identifier or within an identifier are not allowed. The first 32 characters are significant.

Variables can be used everywhere, where allowed by the declared type. You can call the available variables via the input help.

<u>System flags</u>

System flags are implicitly declared variables that depend on their special controls. In order to find out which system flags your system is using, choose the command INSERT OPERAND. In the input help dialogue, go to the category System Variable.

Access to variables from arrays, structures and blocks

Components of two-dimensional arrays can be accessed with the following syntax:

```
<Array name>[Index1, Index2}]
```

Variables of structures can be accessed with the following syntax:

```
<structure name>.<variable name>
```

Variables of function blocks and programs can be accessed with the following syntax:

```
<Function block name>.<Variable name>
```

Addresses

Address

The direct representation of single memory cells are special character strings. These special character strings are created from the concatenation of the percentage character (%), a prefix for the value and one or more natural numbers which are separated from each other by blank spaces.

The following area prefixes are supported:

I	input
Q	output
M	marker

The following prefixes for the size are supported:

X	single bit
None	single bit
B	byte (8 bits)
W	word (16 bits)
D	double word (32 bits)

Examples:

%QX75 and %Q75 Output bit 75

%IW215	input word215
%QB7	output byte 7
%MD48	double word at memorylocation
%IW2.5.7.1	dependent on the control configuration

Whether an address is valid depends on the actual control configuration of the program.

Marker

All supported values can be used to access the marker.

The address %MD48 would e.g. address the bytes 192, 193, 194 and 195 in the marker area (48 * 4 = 192). The first byte is byte no. 0.

Words, bytes and even bits can be accessed in the same way: With %MX5.0 for example, the first bit in the fifth word is accessed. (Usually, bit are saved in words.)

Functions

In ST, also a function call can appear as operand.

Example:

```
Result := Fct(7) + 3;
```

4.2.7 References

IEC 61131 is a standard for industrial automation systems. It comprises 5 chapters and 2 technical reports. By far the most important part of this standard is part 3: Programming Languages.

This part describes a set of high-level programming languages, which can be used for various applications.

Various books deal with IEC 61131-3 standard in detail. To mention some:

- SPS-Standard IEC 1131
  Karl-Heinz John, Michael Tiegelkamp
  ISBN 3-540-58635-0
- SPS-Programmierung mit IEC 1131-3
  Neumann / Grötsch / Lubkoll / Simon
  ISBN 3-486-23348-3
- SPS-Programmierung nach IEC 1131-3
  (zur Einarbeitung in das Thema empfohlen)
  Wellers
  ISBN 3-464-48062-3
- Moderne Programmiertechnik für Automatisierungssysteme
  Peter Wratil
  ISBN 3-8023-1575-8
- Grundkurs IEC 1131
  (Programmierbeispiele für die Prozeßautomatisierung)
  Karl Pusch
  ISBN 3-8023-1807-2

4.3 Special Features of the PacDrive

4.3.1 General Function

The PacDrive - System is a multi-tasking system. In order to avoid access problems to inputs and outputs that are globally available in the system, input and output maps are not created for each task.
The problem is avoided as inputs and outputs are read and written as soon as an instruction, accessing inputs and outputs, is processed.
This kind of processing method also gives a speed advantage.

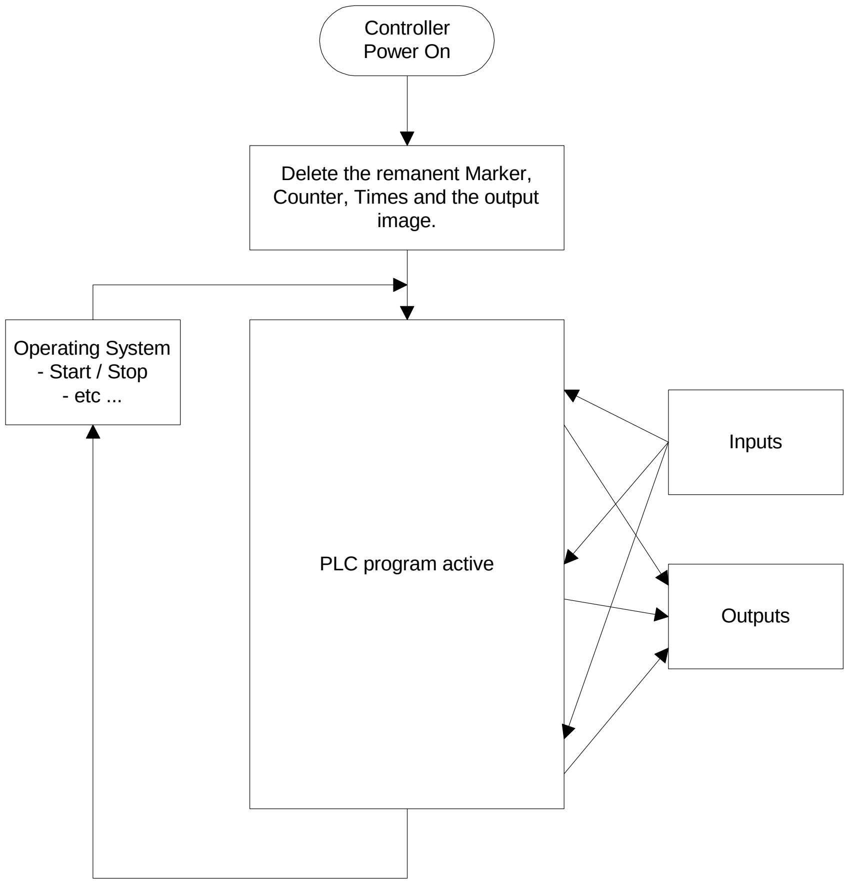


Fig. 4-14: General function of the "multitasking" PacDrive - system

4.3.2 Data Types of the Control Configuration

In the configuration of a PacDrive^{TM} project, data types are declared automatically (see also Programming Manual / Reference / Control Configuration / Table: Objects of the PacDrive - System).

Example:

Transfer of a reference to a block.
An MC-4 was configurated with the object name Axis_10.

```
Procedure(r_MC_4:=Axis_10);
```

In this modular procedure, the logical address can be addressed in the following way:

```
ControllerEnableSet(r_MC_4.logAdr);(* enable for MC-4
  *)
```

4.3.3 Size of Programs and Variable Ranges

See also „System Data“, page 231.

5 Programming Guidelines

5.1 The Concept of Mapping

5.1.1 The Structure of Packaging Machines

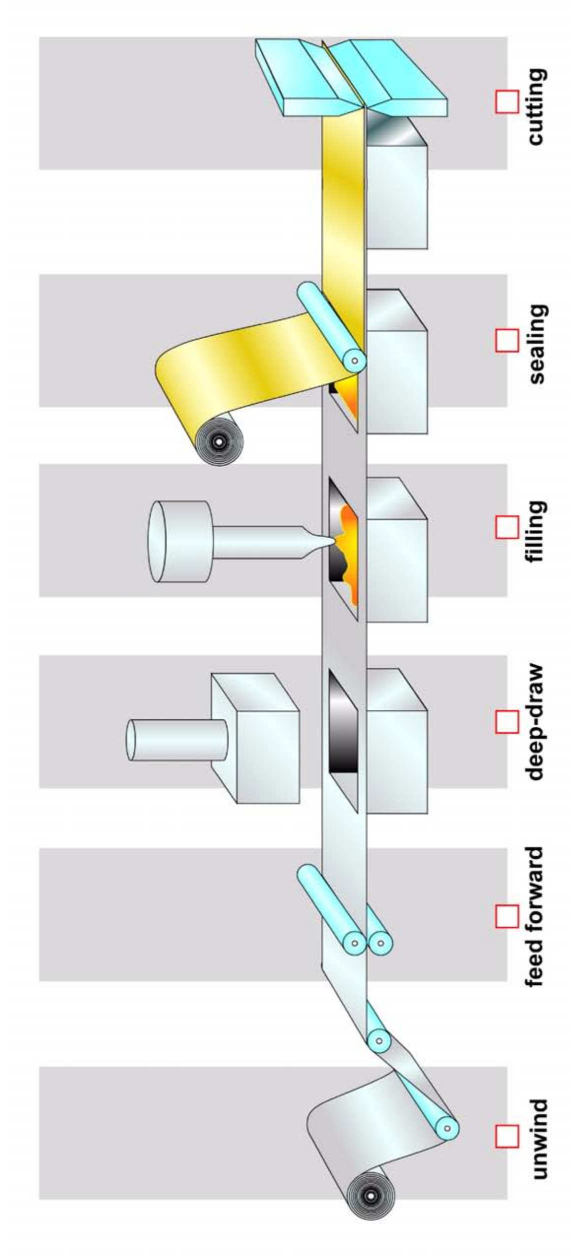


Fig. 5-1: Example of a structure of a packaging machine

5.1.2 The Automation System

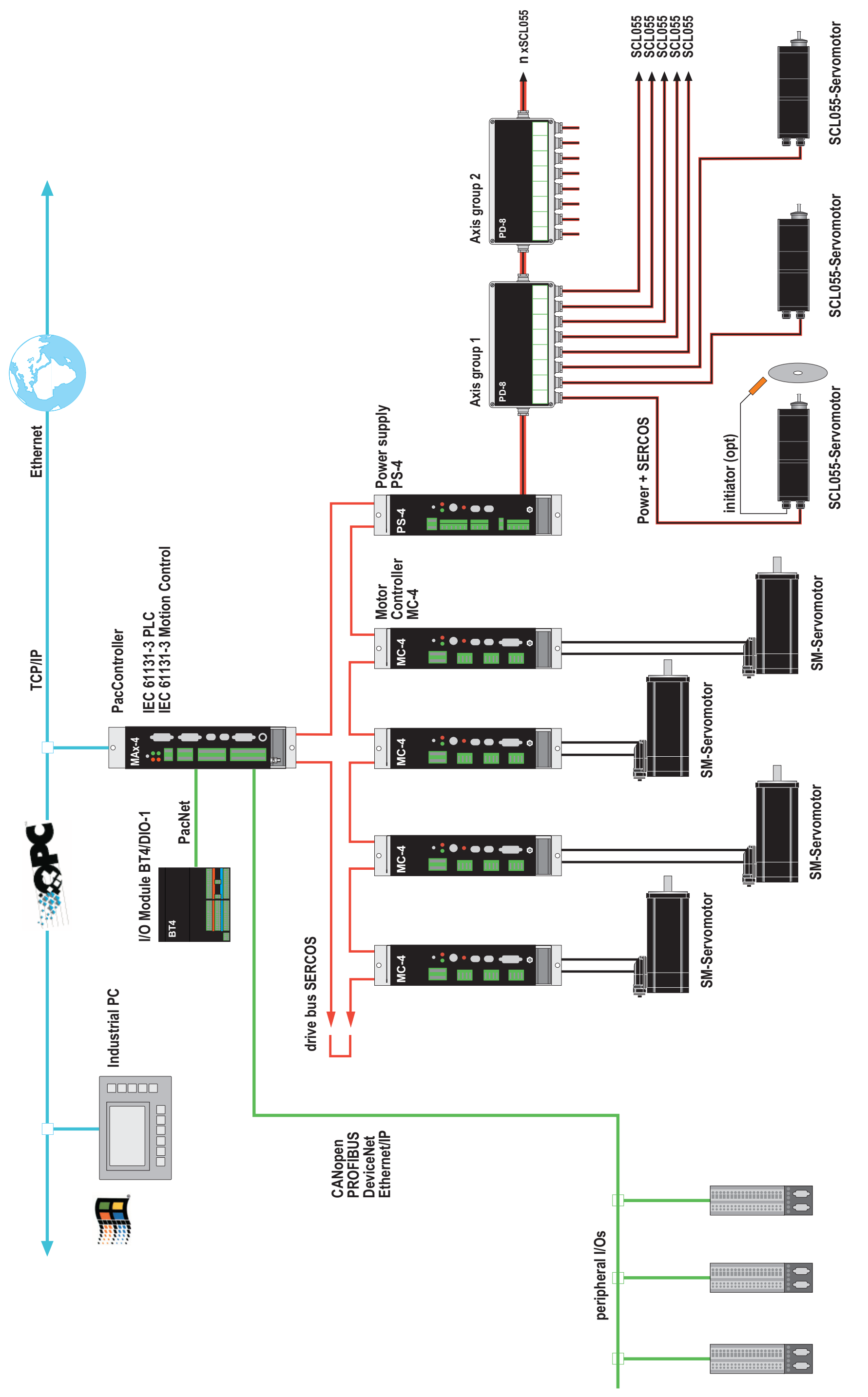


Fig. 5-2: System overview of the PacDrive™ automation system

5.1.3 From Problem Definition to Solution

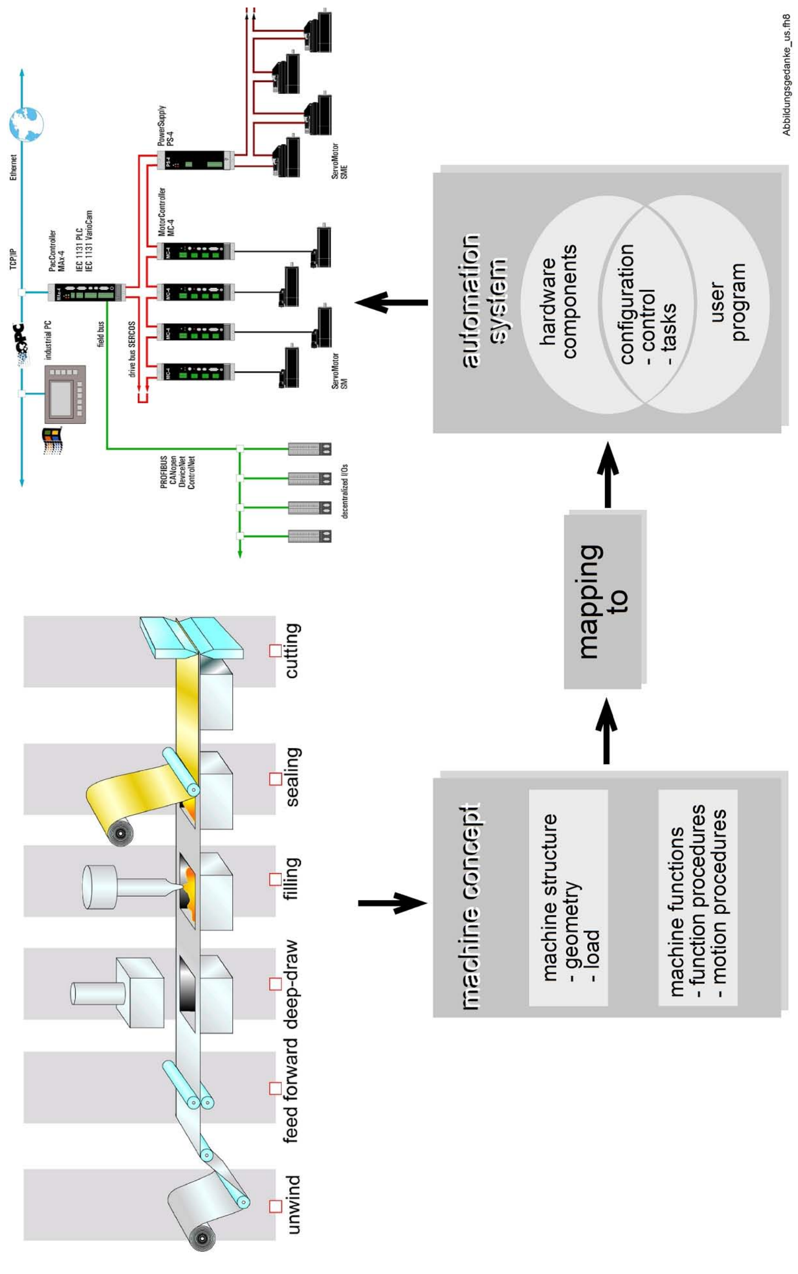


Fig. 5-3: From problem definition to solution with PacDrive

5.1.4 Method for Solving a Defined Problem

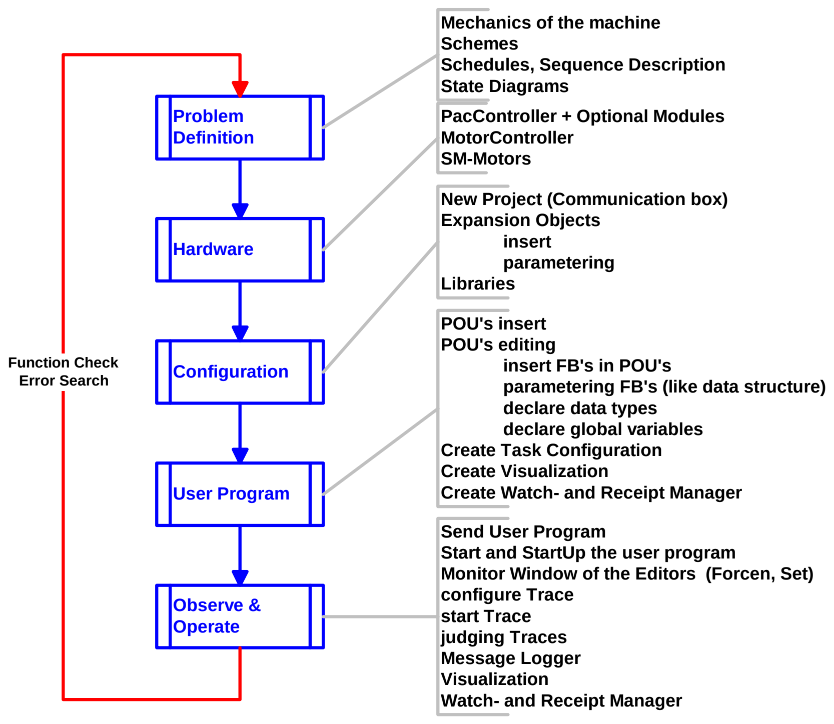


Fig. 5-4: Method for solving a defined problem

5.2 The Structure of a Project

5.2.1 Program Organization with Tasks

In General

As a rule, a PacDrive™ M program (runtime program) consists of several tasks. A task is a program characterized by its priority and cycle time.

Tasks can be used e.g. to determine that a temperature check is only made if there are no more important jobs waiting. Or that the temperature check is made “only“ every 30 seconds.

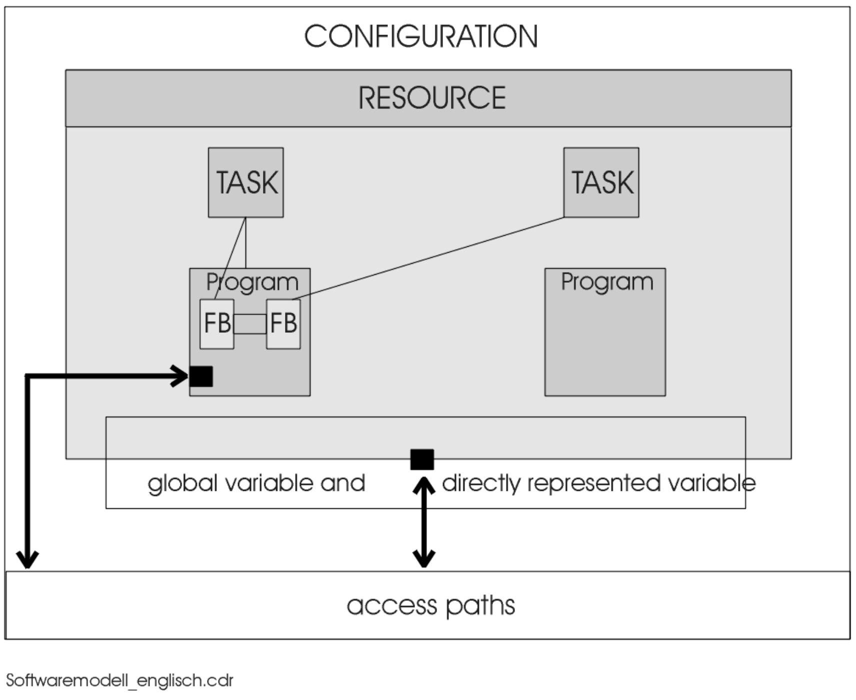


Fig. 5-5: Software model

Characteristics of a task:

- It consists of one or several blocks (POUs; program organization units)
- It has a priority of between 0 and 31 (0 is the highest priority)
- Tasks of higher priority supersede tasks of lower priority
- Tasks of equal priority are processed consecutively in time slices of 250 µs
- A maximum of 127 tasks are supported

The figure below shows how the individual tasks are processed in the system.

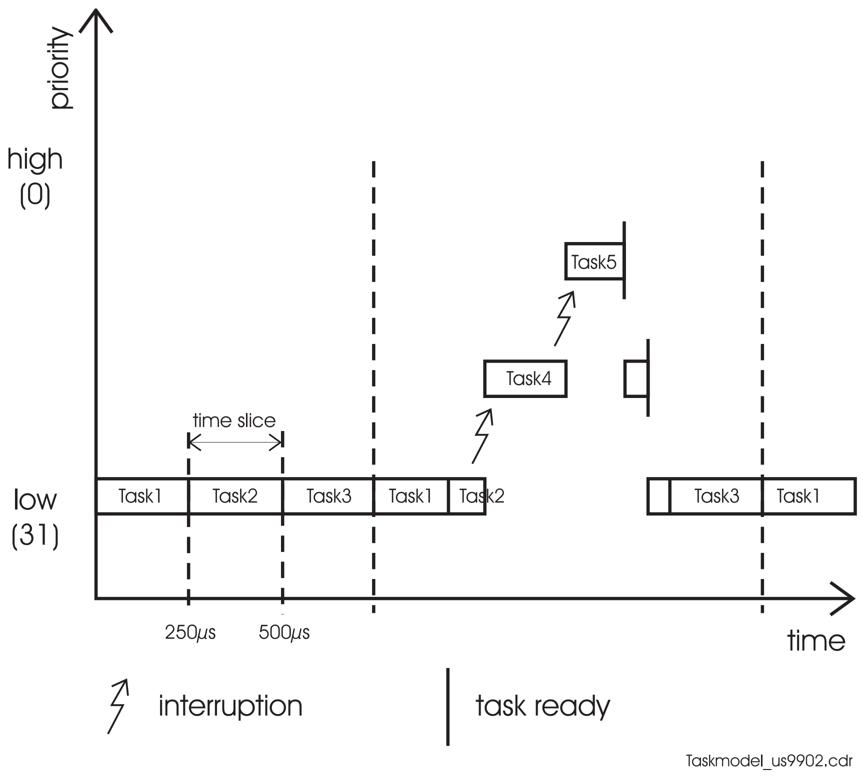


Fig. 5-6: Task model

Tasks of higher priority interrupt tasks of lower priority. After the tasks of higher priority have been processed, the processing of the interrupted tasks of lower priority is resumed.

If there are several tasks of equal priority and they are not interrupted by tasks of higher priority, the tasks of equal priority will be processed in the so-called time slice method. This means that each task is processed for 250 µs before the next task is processed for another 250 µs.

Overleaf you will find a practical example for a task processing.

1) Task change due to higher priority

2) Task change at the end of a time slice in case of tasks of equal priority

3) Task change due to higher priority

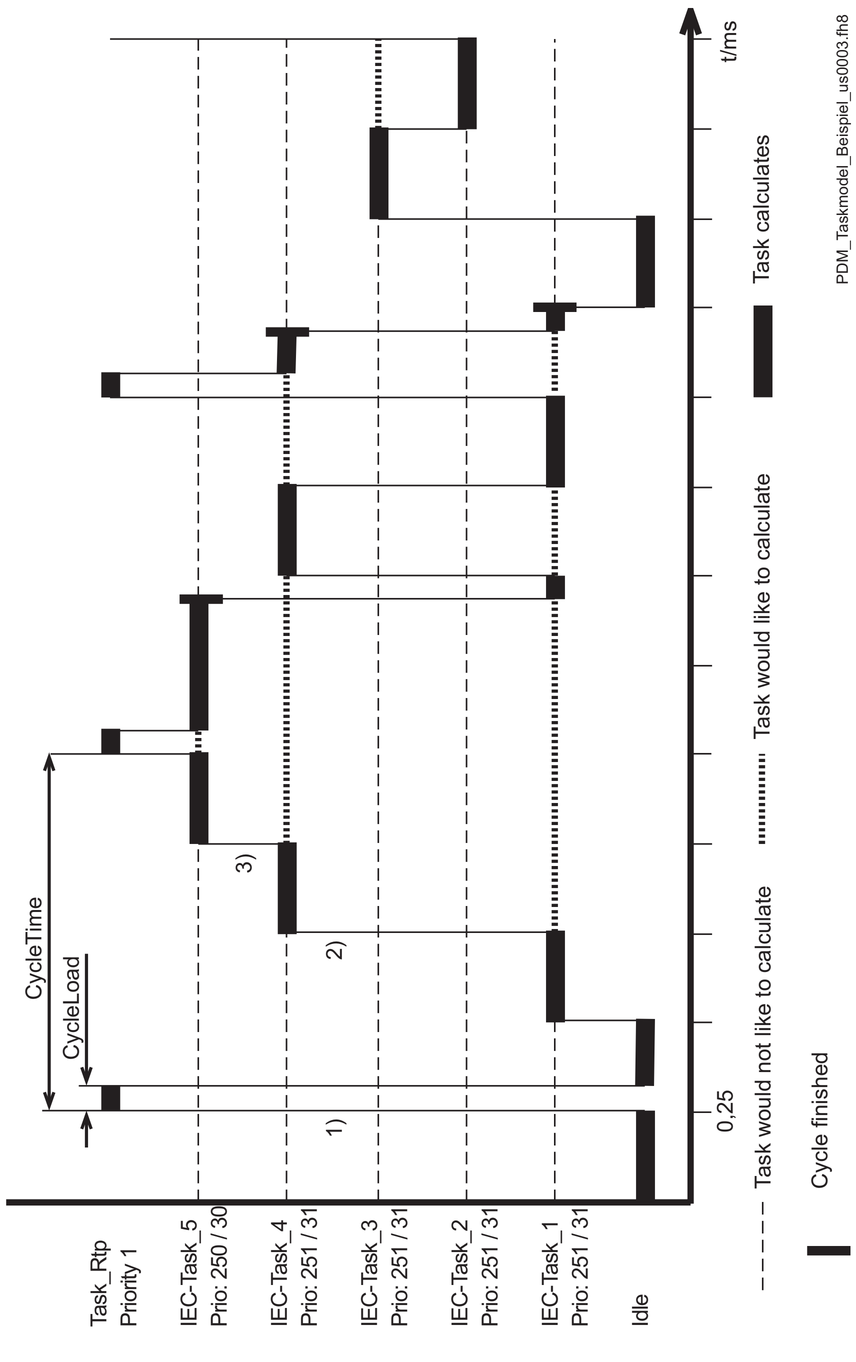


Fig. 5-7: Practical example for a task model

The following figure shows the connection between the set interval for a task, the actual interval time and the load time of the task. For this figure, the following values apply:

- set interval: 10ms
- cycle is the actual interval time
- load is the processing time or load time of the task

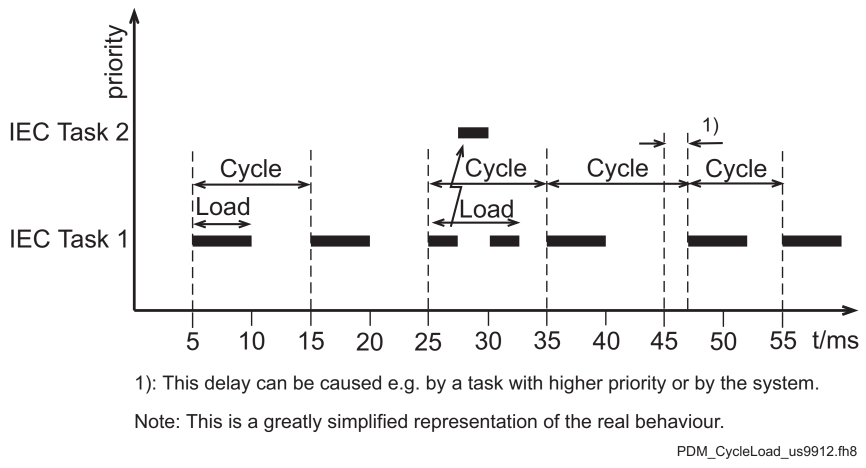

priority
IEC Task 2
IEC Task 1
Cycle
Load
Cycle
Load
Cycle
Cycle
1)
5
10
15
20
25
30
35
40
45
50
55
t/ms
1): This delay can be caused e.g. by a task with higher priority or by the system.
Note: This is a greatly simplified representation of the real behaviour.
PDM_CycleLoad_us9912.fh8

Fig. 5-8: Example for the behavior of cycle and load

The system tries to adhere to the set interval. It may happen, however, that an interval is longer than the given time. This is the case if the new start of a cycle is delayed by the system or a task of higher priority. The system then tries to shorten the next cycle in order to return to the time grid.

The processing of the programs allocated to a task is marked by “load“. The processing of the programs must be completed within the set interval; otherwise there will be a diagnosis message (cycle time transgression or serious cycle time transgression).

NOTE

There are several system tasks which are processed according to the same principles. You will find those system tasks and the classification of their IEC priority later in this manual under the heading “Which is the right priority?“.

Developing a project-specific task model

At the beginning of a project, you need to develop a task model. The following points need to be defined:

- number of tasks
- priority of the tasks
- cycle time

NOTE

Parameters that can be read and/or written via SERCOS may delay program execution significantly (e.g. for TRACE, Error Quit, ...). See also EPAS-4 Online help / Control Configuration / Types of Object Parameters

---

How many tasks do I need?

In many cases it is sufficient to use one single task for the positioning function or the entire project. In addition, further tasks may be required for monitoring functions, PLC functionality, operating units, etc. The aim is to use as few tasks as possible.

General rules:

- If possible there should be only one task.
- Functions that take a longer time to process (e.g. file services, access to SERCOS parameters) should be moved to a task of lower priority.
- It may be necessary to introduce a task of higher priority e.g. for safety reasons.
- In practice you rarely need more than three tasks.
- There should be no tasks of equal priorities. In this case, it is better to call up the functions from a “jump-in program“. This results in a fixed time frame for the respective tasks.

Which is the right priority?

The PacDrive™ system has 252 priorities internally. The lowest 32 of these priorities are available for the IEC programs. From the figure below, you can see the allocation of the IEC priority in the system priority.

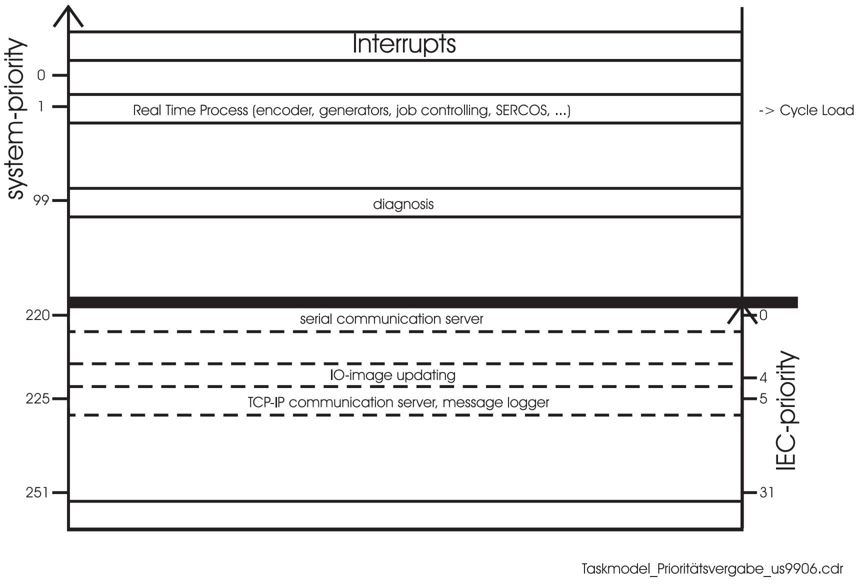

Interrupts
system-priority
0
1
Real Time Process (encoder, generators, job controlling, SERCOS, ...)
-> Cycle Load
99
diagnosis
220
serial communication server
0
IO-image updating
4
225
TCP-IP communication server, message logger
5
IEC-priority
251
31
Taskmodel_Prioritätsvergabe_us9906.cdr

Fig. 5-9: Task model / priority allocation

These rules apply to the user program:

- If only one task is used, the priority of that task hardly matters. you should choose a priority between 6 and 31.
- A task of lower priority may be used for functions which require e.g. a longer time for processing (e.g. file services, access to SERCOS parameters) or whose execution may be superseded (maybe heating regulators).
- It may be necessary to introduce a task of higher priority for safty aspects.
- If the cycle time of a task must be less than 5ms, you should choose a priority of less than 5 in order to avoid interruptions by the TCP-IP communication server.

PD_UserMan_Richtlinien_us.fm

NOTE

Should sporadic cycle errors (505 “simple cycle error“, 506 “multiple cycle error“) occur in your application, the reason may be that the Ethernet communcation (Ftp, Ping, EPAS-4, ...) slows down the processing of your IEC task.
In this case it is advisable to choose the IEC task priority higher than 5 (priority at 0 ... 4).
If the priority of an IEC task accessing field bus modules is higher than (0 ... 3) or equal to (4) the priority of the IO map updating, data consistency is only ensured for an element access (e.g. Byte, Bool, Word, ...).

<u>Which cycle time should I choose?</u>

For the selection of the cycle time, observe the following points:

- cycle time of the positioning procedure
- minimum time in which a CAM function must be processed
- access to SERCOS parameters (delay! -> see EPAS-4 Online help / Control Configuration)
- use of functions with delay

NOTE

A typical value for the cycle time lies in the range of 5 ... 10 ms.
If a cycle time of less than 5ms must be achieved, it may be necessary to change the priority of the task.
-> see also “Which is the right priority“?

Monitoring

<u>What types of monitoring are there?</u>

The execution of cyclic tasks is monitored. The following diagnosis messages can occur:

Class (default)	Diagnosis code	Language-specific diagnosis text	quittable (default)
4	313	serious cycle time transgression	no
6	317	cycle time transgression	yes

<u>What are the causes and how can they be cleared?</u>

313 Serious cycle time transgression	
The serious cycle time error of an IEC task is reported if the 10-fold cycle time is exceeded (see Task Configuration). The task moves to error state. This state can only be quitted by an IEC task reset. Note: The cycle time monitoring of the IEC tasks refers to the object parameter Load, which is shown in the expansion object IEC Task.	
Cause 1: Handling:	Call of a parameter or system component which takes “very long“ to process. Check program (e.g. access to SERCOS parameter -> see Types of Object Parameters or Function WAITT).
Cause 2: Handling:	Time interval too short. Check task configuration.

317 Cycle time transgression	
A "minor" cycle time error was detected. It is triggered if the cycle time is exceeded by more than 250µs. The actual cycle time / cycle time from the task configuraiton in ms can be found in DiagExtCode.	
Cause: Handling:	See diagnosis message 313

FastTask concept

The “FastTask“ concept permits high-priority, fast, synchronous and interrupt-controlled IEC tasks.

Further information on the FastTask concept is available in EPAS-4 online help > library “MAx-4“ > IEC_Tasks > General notes on the FastTask concept.

NOTE

Before you use the “FastTask“ concept, we recommend that you familiarize yourself thoroughly with the subject and contact our application department.

IEC task state machine

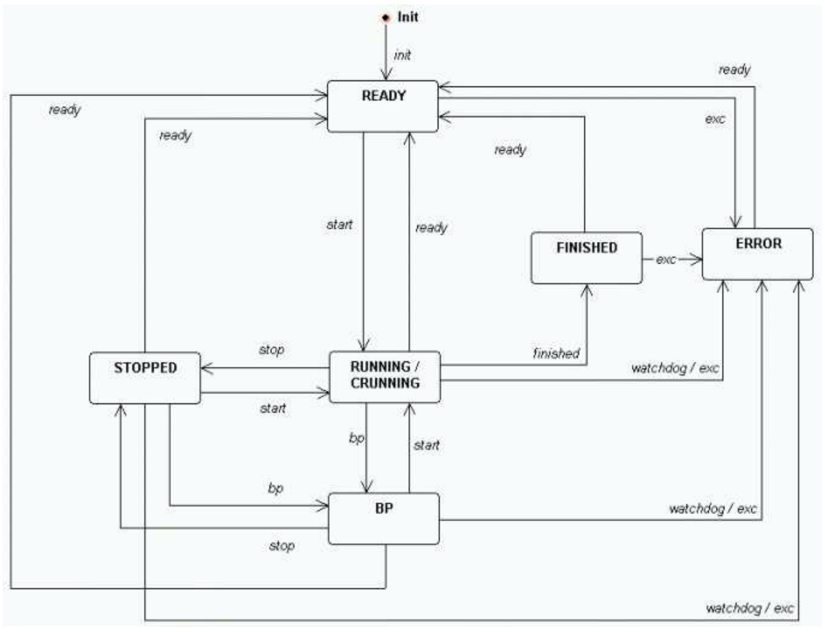


Fig. 5-10: IEC task state machine

The state of the task can be read with the help of the configuration object IEC Task (Parameter State).

Value	Designation	Meaning
0		no task initialization state; no task defined
1	READY	ready task defined and ready to start
2	RUNNING	running task is running

PD_UserMan_Richtlinien_us.fm

Value	Designation	Meaning
3	CRUNNING	cyclic operation task is running in cyclic operation
4	STOPPED	stopped task stopped
5	BP	standing at breakpoint task stopped by a breakpoint or single-step
6	ERROR	error a serious error occurred; task suspended
7	FINISHED	finished task has reached its end

Table 5-1: Values of the IEC state machine

Measuring possibilities

It is possible to measure the capacity utilization of the IEC tasks. For this purpose you can use the IEC task object, which you can insert in the control configuration. The individual parameters are described in the EPAS-4 Online help / Control Configuration.

Method:

- Insert as many IEC task objects in the control configuration as you have tasks entered in the task configuration.

NOTE

The first IEC task object in the control configuration is linked with the first task in the task configuration. The first n IEC task objects are linked with the first n tasks in the task configuration. In principle, there can be more or less IEC task objects in the system than there are tasks.

- Transmit the project to the PacDrive Controller.
- To start measuring, set EnableLoad to on / TRUE.

5.3 ELAU Program Structure

To create a program structure, it is important for you to know in which areas the blocks (POUs = program organization units) differ. See also „The Basics of IEC-61131“, page 25.

Create the first level of your program in the programming language SFC (sequential function chart).

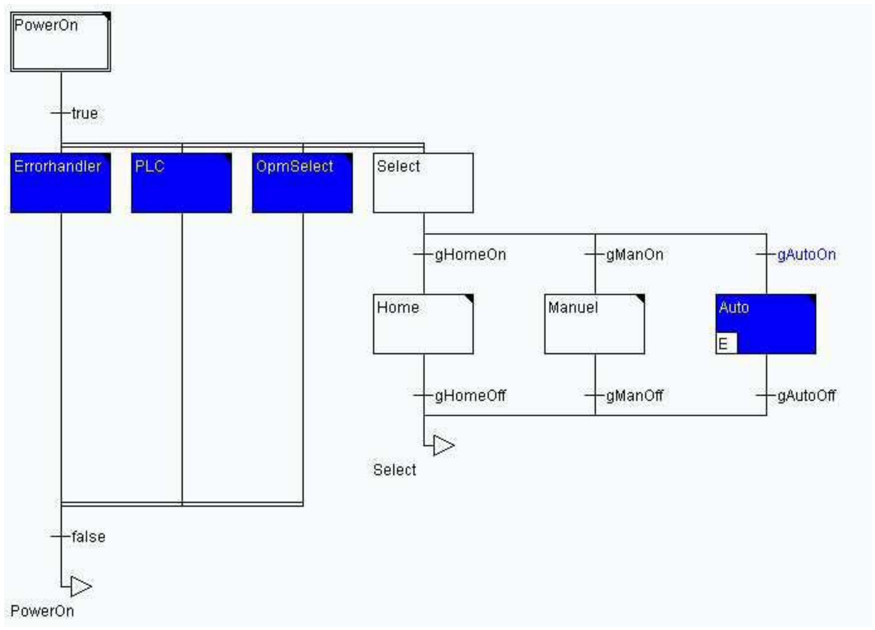


Fig. 5-11: Example for the first level in SFC

The inferior program parts should be created in FBD.

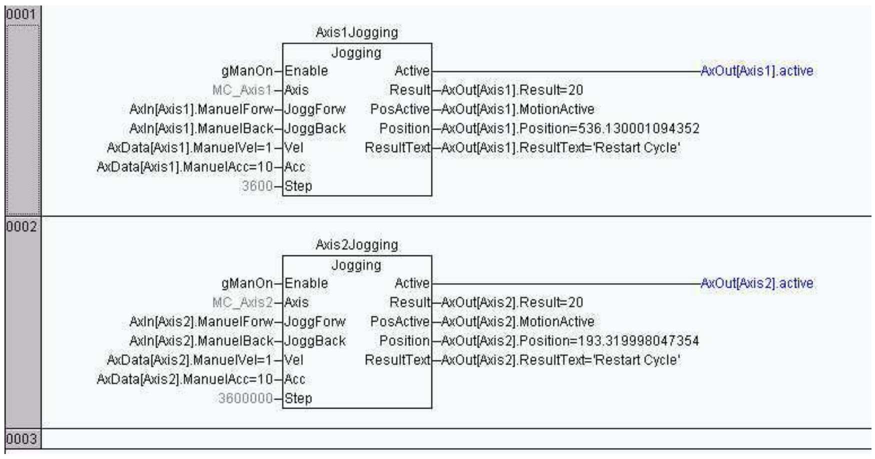
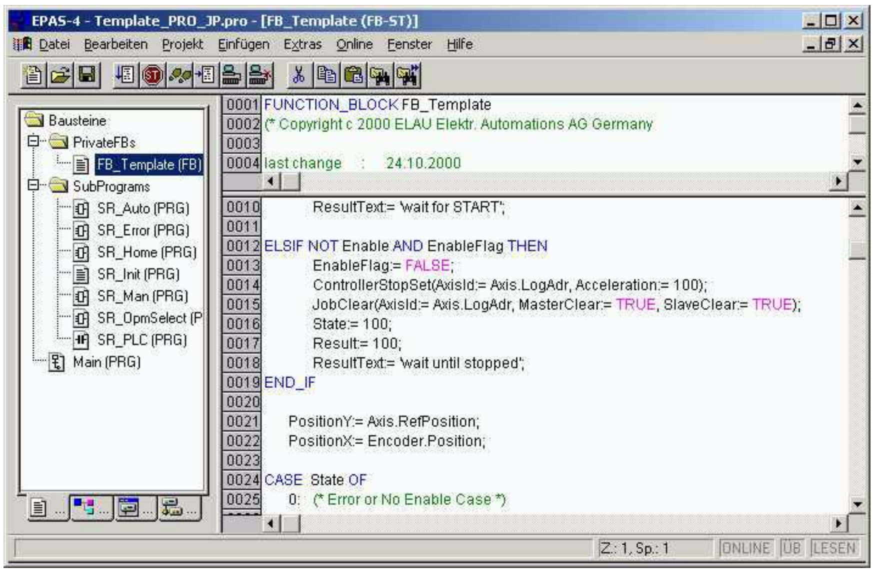


Fig. 5-12: Example for an inferior program in FBD

More complex functions and function blocks should be programmed in ST, as this programming language offers comfortable commands for structuring (e.g. WHILE, FOR, CASE, ...).



```
FUNCTION_BLOCK FB_Template
(* Copyright c 2000 ELAU Elektr. Automations AG Germany

last change    :     24.10.2000
```

```
        ResultText:= 'wait for START';

ELSIF NOT Enable AND EnableFlag THEN
        EnableFlag:= FALSE;
        ControllerStopSet(AxisId:= Axis.LogAdr, Acceleration:= 100);
        JobClear(AxisId:= Axis.LogAdr, MasterClear:= TRUE, SlaveClear:= TRUE);
        State:= 100;
        Result:= 100;
        ResultText:= 'wait until stopped';
END_IF

    PositionY:= Axis.RefPosition;
    PositionX:= Encoder.Position;

CASE  State OF
    0:   (* Error or No Enable Case *)
```

Fig. 5-13: Example for a function block in ST

Use of state machines

In practice, it has turned out to be sensible, and in most cases necessary, to build the programming in the programs, functions and function blocks on state machines.

When programming in structured text (ST), state machines are created with the CASE command.

```
CASE lState OF
      1:     (* initialization *)
             < commands >
             lState:=lState+1;(* switch to next state *)


      2:     (* state 2 *)
             < commands >
             lState:=lState+1;(* switch to next state *)
      :
      n:     (* state n *)
             < commands >
             lState:=x;    (* switch to next state*)
END_CASE;
```

5.3.1 ELAU Functions and Function Blocks

ELAU offers a number of libraries with functions and function blocks focusing on certain subjects.

- Standard.lib
  Standard IEC library with elementary functions (e.g. counter, string functions, timer, etc.).
- MAx-4.lib
  Interface library for basic functions of MAx-4 PacController.
- Check.lib
  Interface library for checking ARRAYs (debugging functions).
- Basic.lib
  IEC library with positioning functions. Based on MAx-4.lib.
- IECsfc.lib
  Use of IEC steps with V00.06.00 or higher.
- CSpline.lib
  Interface library on VarioCam® CSpline functions for online conversion of supporting point data into cam data.
- TorqueCam.lib
  Interface library on torque cams.
- CANL2.lib
  Interface library on CAN Layer 2.
- CANopen_M.lib
  Interface library on CANopen Master
- CANopen_S.lib
  Interface library on CANopen Slave.
- PBDP_M.lib
  Interface library on PROFIBUS-DP Master.
- PBDP_S.lib
  Interface library on PROFIBUS-DP Slave.
- DNet_S.lib
  Interface library on Device Net Slave
- HMI_pcs.lib
  Interface library for connection with Lauer PCS
- HMI_Simatic.lib
  Interface library for connection with Simatic.
- ModBus.lib
  Interface library for connection with ModBus.
- TSC.lib
  Interface library for connection with ModBus (see also Operating Manual HMI Libraries).

- VarioCam.lib
  Library for complex positioning tasks.
- PLC Library
  Bit functions, communication, PID regulator, conversion functions, etc.

Further libraries (separate products)

- robotic library
- library for robot applications

Runtime properties

The processing times of the functions and function blocks differ greatly. Here are some examples:

- MultiCam3(): typically 12 µs
- AppendToWriteFile(): several 100 ms

To be able to use blocks with longer processing time, it is often necessary to increase the cycle time monitoring of a task temporarily. For this purpose, you can use the function *CycleCheckTimeSet()*.

5.3.2 Parameter Transmission

For parameter transmission, please note:

VAR_IN / VAR_OUT

Transmission of the value, i.e. the value is copied. This means that the transmission of the value requires memory space and time to copy the value to the new place.
Therefore avoid transmitting large variables (ARRAY and STRUCT).

NOTE

Data types of the control configuration (MC_4, D_IN, ...) are automatically transmitted as a reference.
In case of a transmission for reference, the variables in the program seem to be transmitted “normally“. Internally, however, no copy of the variable, but only a pointer is transmitted.

VAR_IN_OUT

Transmission to the address of the variable.
The advantage is that no copy procedure is initiated, which saves resources and time.

5.4 Designator Names

The purpose of the following project conventions is to standardize the names for objects, variables and instances of objects and variables. The code and particularly the interfaces of function blocks and functions thus are easy to read and understand. Code maintenance is facilitated.

This should be regarded as a guideline for PacDrive^{TM} users and programmers, as IEC 61131-3 does not prescribe those programming conventions.

The designators should all be named with the same prefixes, so that the object type can easily be identified.

Prefix	Type	Example
TASK_	Task	TASK_Main
FB_	Function block	FB_Pos
ST_	Structure	ST_BufferEntry
E_	Enumeration type (ENUM)	E_PosStates
T_	Reference (TYPE)	T_Nibble
P_	Program	P_Main
F_	Function	F_Convert

Table 5-2: Defining object names

Prefix	Type	Example
fb	Instance name of a function block	fbPos
st	Instance name of a structure	stBufferEntry
e	Instance name of an enumeration type (ENUM)	ePosStates
none	Instance of a reference type	Nibble

Table 5-3: Generate the instances of the object names

Give your variables “talking“ names.

We recommend two prefixes:

- scope prefix
- type prefix

Scope prefix	Type	Example
no prefix	local variable	bName

Scope prefix	Type	Example
I_	input	I_bSleighThere
O_	output	O_bValveOpen
g_	global variable	g_bName
s_	static variable (constant)	s_bName
r_	reference (e.g. to object MC_4)	r_MC_4

Table 5-4: Scope prefixes

Type prefix	Type	Keywords	Example
b	BOOL	boolean (1)	bName
c	BYTE	bit string 8 (8)	cName
w	WORD	bit string 16 (16)	wName
lw	DWORD	bit string 32 (32)	lwName
si	SINT	short integer (8)	siName
i	INT	integer (16)	iName
l	DINT	double integer (32)	lName
usi	USINT	short integer (8)	usiName
ui	UINT	integer (16)	uiName
ul	UDINT	double integer (32)	ulName
r	REAL	floating-point number (32)	rName
lr	LREAL	long floating-point number (64)	lrName
d	DATE	date (32)	dName
to	TOD	time (32)	toName
dt	DT	date and time (32)	dtName
t	TIME	duration (32)	tName
s	STRING	string	sName
p	Pointer	pointer	pbName
a	Array	field	abName
t	Struct	structure	tbName

Table 5-5: Type prefixes

5.5 Upper and Lower Case

The prefix in object names is always in upper case. The separator between prefix and object name is an underscore “_“. The prefix for variable and instance names is always in lower case. The first character of a designator is always in upper case. If a designator consists of several words, the first character of a word is always in upper case. No separator (e.g. _) should be used between words.

NOTE

EPAS-4 is not case-sensitive.

5.6 Valid Characters

The designators should only contain the following letters, numbers and special characters:

0 ... 9, A ... Z, a ... z

NOTE

All designators should be in English.

5.7 Create a PacDrive^{TM} project

5.7.1 Problem Definition

We want to create a simple project. The only requirement is that the output of the MAx-4 flashes at a frequency of 1 Hz.

PD_UserMan_Richtlinien_us.fm

Define input and output variables, sensor and actuators

Opera-ting means	Input/output designation	Comment
H1	O_bEquipmentOn-IndicatorH1	An indicator lamp is connected to the output. It will flash at a frequency of 1 Hz as soon as the equipment is switched on.

Table 5-6: I/O definition

NOTE

Unclear function descriptions result in misinterpretations. Therefore work with particular diligence at this point. Supplement the description by diagrams where necessary.

Identifier	Kind of operating means	Examples
A	assembly groups	
B	converter from non-electrical to electrical sizes and vice-versa	measuring converter for temperature, light, turning frequency, etc. approximation initiators, way and angle converters
C	capacitors	
D	binary elements, deceleration facilities, storage facilities	
E	various	
F	protection means	
G	power supplies, generators	
H	messaging units	
K	contactors, relays	
L	inductivities	
M	motors	
N	amplifiers, regulators	
P	measuring units, test facilities	

Identifier	Kind of operating means	Examples
Q	high-voltage switch gear	
S	switches, dials	calipers, limit tracers, command units, dials
T	transformer	
U	modulator, converter from electrical sizes to other electrical sizes	
V	semiconductors, tubes	
W	transmission ways, lines, aerials	
X	clamps, connector plugs, sockets	
Y	electronically actuated mechanical facility	brakes, clutches, valves
Z	filters, equalizers, limiters, terminations	

Table 5-7: Identification of electrical operating means

Depict the development over the time

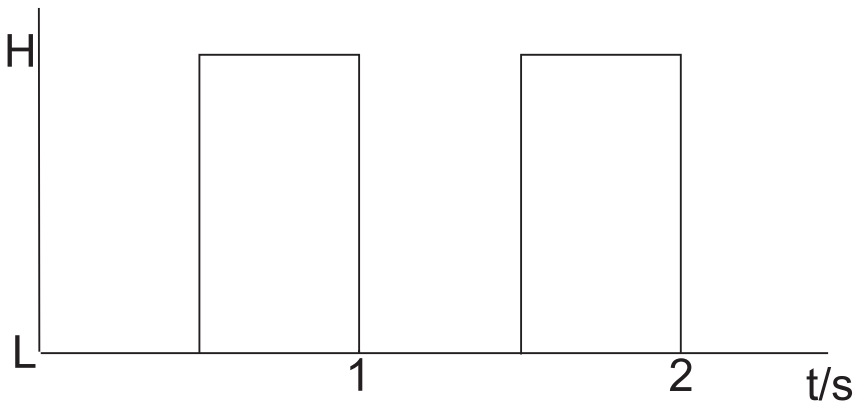


Fig. 5-14: Development over the time

5.7.2 Hardware

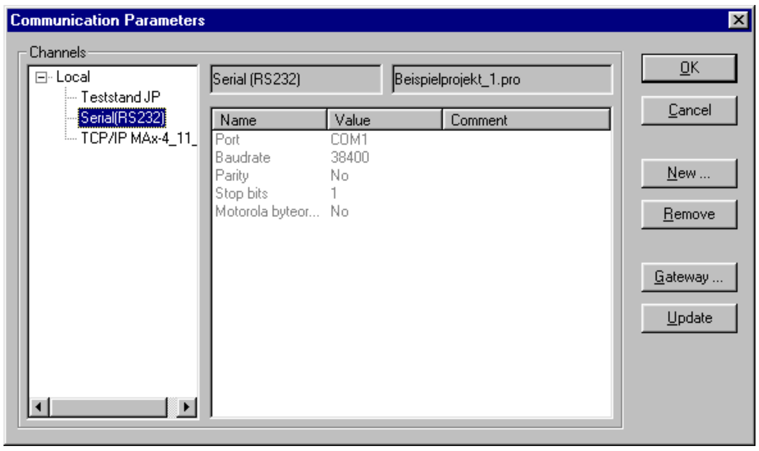
A MAx-4 PacDrive Controller is sufficient to solve this problem.

5.7.3 Configuration

Create a project

➢ To create a new project, click on FILE | NEW.

You will see a dialog box where the parameters for the online connection are defined.
(-> for further information, see Operating Manual EPAS-4)

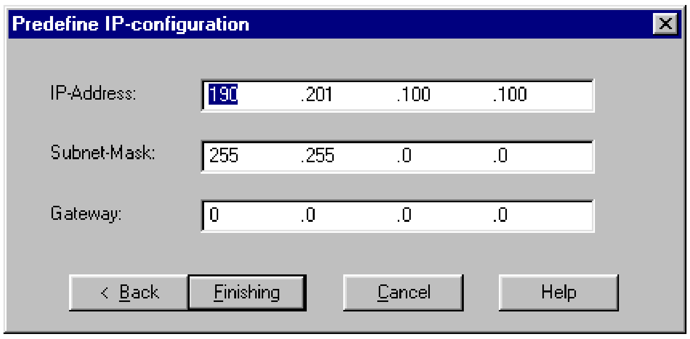

Communication Parameters

Channels
Local
Teststand JP
Serial(RS232)
TCP/IP MAx-4_11_

Serial (RS232) | Beispielprojekt_1.pro

Name	Value	Comment
Port	COM1	
Baudrate	38400	
Parity	No	
Stop bits	1	
Motorola byteor...	No	

Fig. 5-15: Communication parameter box


Predefine IP-configuration

IP-Address: 190 .201 .100 .100

Subnet-Mask: 255 .255 .0 .0

Gateway: 0 .0 .0 .0

Fig. 5-16: IP address assistant

Configure a project

The composition of the PacDrive™ system and the “properties“ of the individual parts are defined in the control configuration.

PD_UserMan_Richtlinien_us.fm

A PacDrive™ system consists of at least one MAx-4 PacDrive Controller, which basically provides the following configuration groups.

Group	Explanation
general	general parameters of thePacDrive Controller
diagnosis	diagnosis parameters of the PacDrive Controller
versions	version parameters of the PacDrive Controller
real-time bus	object of the PacDrive Controller
digital output groups	object of the PacDrive Controller
digital input groups	object of the PacDrive Controller
digital measuring input groups	object of the PacDrive Controller

Table 5-8: Basic configuration groups / objects

➢ Select *Resources* to call up the configuration editor.

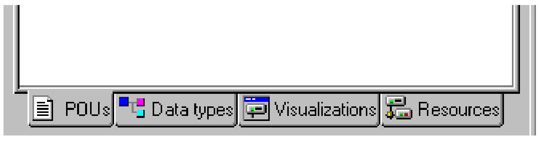


Fig. 5-17: Selection of resources in the Object Organizer

➢ Now select the item *Control configuration*.

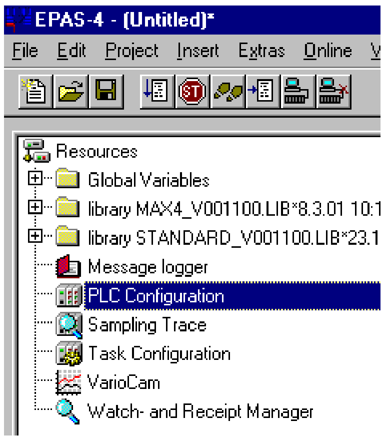
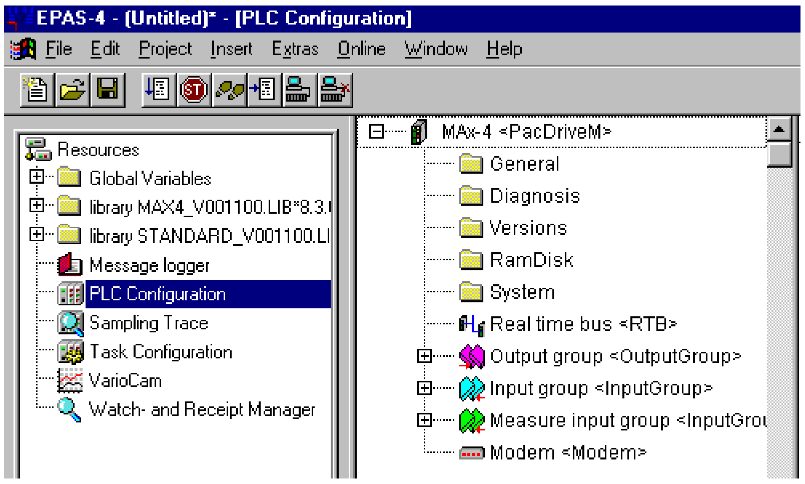

EPAS-4 - (Untitled)*
File Edit Project Insert Extras Online V
Resources
Global Variables
library MAX4_V001100.LIB*8.3.01 10:1
library STANDARD_V001100.LIB*23.1
Message logger
PLC Configuration
Sampling Trace
Task Configuration
VarioCam
Watch- and Receipt Manager

Fig. 5-18: Select control configuration

The minimum configuration for a PacDrive™ system will appear. For our first example, this is sufficient.


EPAS-4 - [Untitled]* - [PLC Configuration]

File Edit Project Insert Extras Online Window Help

Resources
- Global Variables
- library MAX4_V001100.LIB*8.3.
- library STANDARD_V001100.LI
- Message logger
- PLC Configuration
- Sampling Trace
- Task Configuration
- VarioCam
- Watch- and Receipt Manager

MAx-4 <PacDriveM>
- General
- Diagnosis
- Versions
- RamDisk
- System
- Real time bus <RTB>
- Output group <OutputGroup>
- Input group <InputGroup>
- Measure input group <InputGrou
- Modem <Modem>

Fig. 5-19: Control configuration with minimum configuration

Two files for the control configuration, the *configuration file* and the *parameter file* are stored in the PacDrive Controller.

In principle, the objects (MC-4 MotorController, logic encoder, etc.) are administered in the *control configuration*.
The “inputs with user initialization“ (see “Types of MAx-4 PacDrive Controller object parameters“) are stored with their initial values in the *parameter file*.

NOTE

If the *control configuration file* is changed (object added or deleted), the PacDrive Controller must be reset (or booted)!
For further information, please see Programming Manual -Reference- , chapter Control Configuration.

In principle, we do not need to change the default configuration for our project. However, we want to give our output a “sensible“ symbolic name. We already defined this name when defining the problem.

➢ To enter the name, click on the symbolic name written in brakkets after output_0.

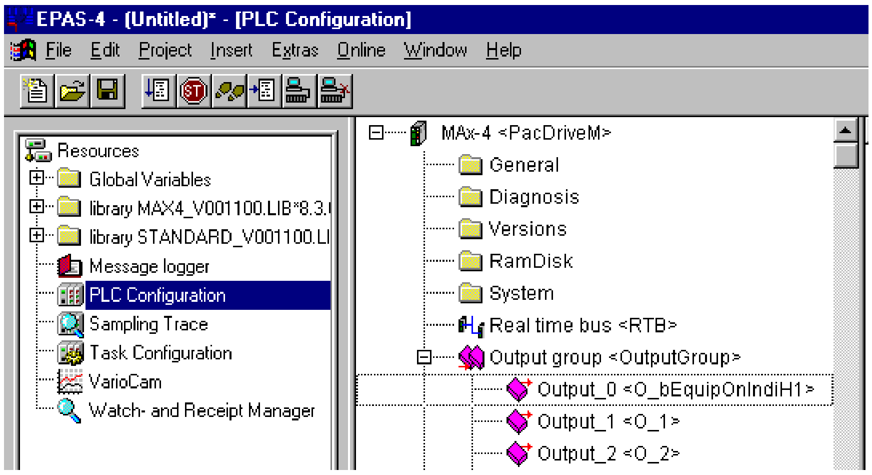

EPAS-4 - (Untitled)* - [PLC Configuration]
File Edit Project Insert Extras Online Window Help
Resources
Global Variables
library MAX4_V001100.LIB*8.3.
library STANDARD_V001100.LI
Message logger
PLC Configuration
Sampling Trace
Task Configuration
VarioCam
Watch- and Receipt Manager
MAx-4 <PacDriveM>
General
Diagnosis
Versions
RamDisk
System
Real time bus <RTB>
Output group <OutputGroup>
Output_0 <O_bEquipOnIndiH1>
Output_1 <O_1>
Output_2 <O_2>

Fig. 5-20: Enter symbolic name for output_0

5.7.4 User Program

➢ To be able to enter the program, please switch to Components.

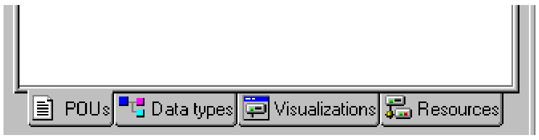


Fig. 5-21: Component selection in the Object Organizer

➢ Click with the right mouse button on “POUs“ and then select “Add object...“.

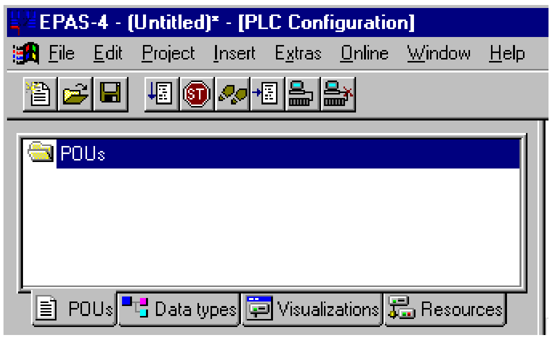
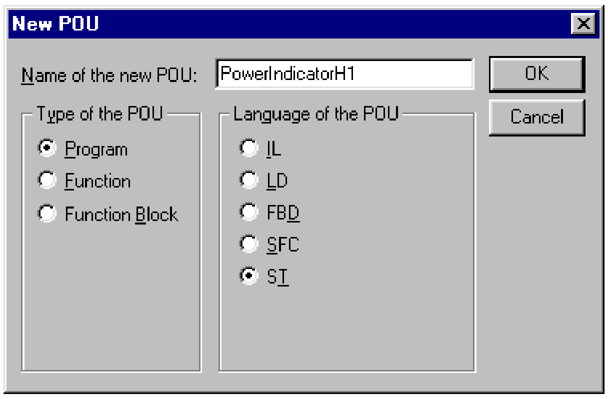

EPAS-4 - (Untitled)* - [PLC Configuration]
File Edit Project Insert Extras Online Window Help
POUs
POUs
Data types
Visualizations
Resources

Fig. 5-22: Click on POUs with the right mouse button

➢ Now enter a talking name for your program and confirm with OK.



New POU

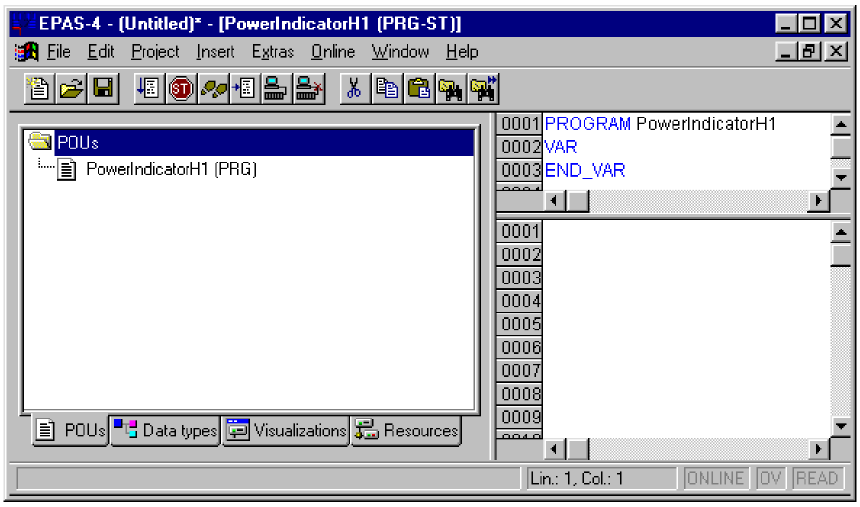
Name of the new POU: PowerIndicatorH1

Type of the POU: [✓ Program] Function Function Block

Language of the POU: IL LD FBD SFC [✓ ST]

Fig. 5-23: Add component / dialog box

You are now in program editor for the programming language *Structured Text*.

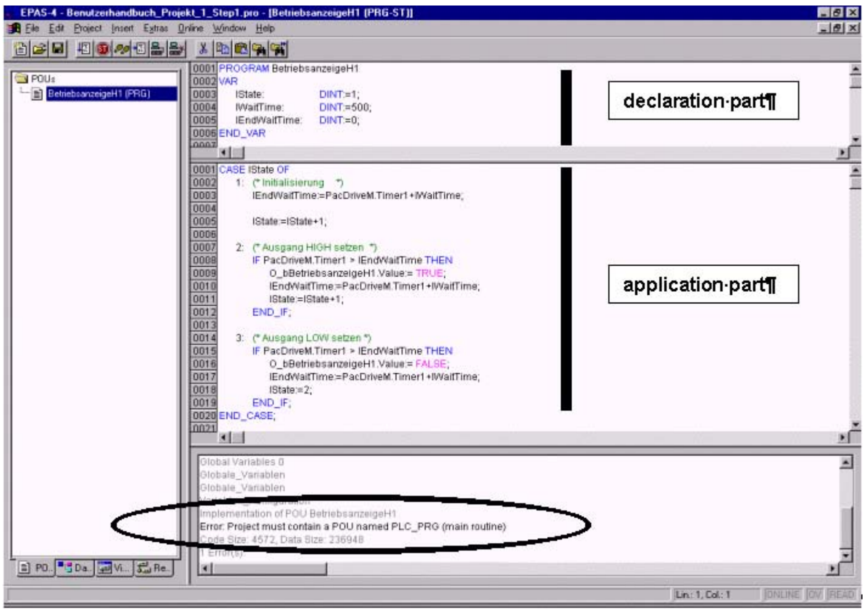


EPAS-4 - (Untitled)* - [PowerIndicatorH1 (PRG-ST)]

```
PROGRAM PowerIndicatorH1
VAR
END_VAR
```

Fig. 5-24: Program editor for Structured Text

➢ Now enter the program and compile it by pressing F11 (or MENU | PROJECT | COMPILE ALL).


EPAS-4 - Benutzerhandbuch_Projekt_1_Step1.pro - [BetriebsanzeigeH1 (PRG-ST)]
File Edit Project Insert Extras Online Window Help
POUs
BetriebsanzeigeH1 (PRG)
declaration·part¶
application·part¶
PO.. Da.. Vi... Re..
Lin.: 1, Col.: 1
ONLINE OV READ

```
PROGRAM BetriebsanzeigeH1
VAR
    lState:         DINT:=1;
    lWaitTime:      DINT:=500;
    lEndWaitTime:   DINT:=0;
END_VAR
```

```
CASE lState OF
    1:  (* Initialisierung    *)
        lEndWaitTime:=PacDriveM.Timer1+lWaitTime;

        lState:=lState+1;

    2:  (* Ausgang HIGH setzen  *)
        IF PacDriveM.Timer1 > lEndWaitTime THEN
            O_bBetriebsanzeigeH1.Value:= TRUE;
            lEndWaitTime:=PacDriveM.Timer1+lWaitTime;
            lState:=lState+1;
        END_IF;

    3:  (* Ausgang LOW setzen *)
        IF PacDriveM.Timer1 > lEndWaitTime THEN
            O_bBetriebsanzeigeH1.Value:= FALSE;
            lEndWaitTime:=PacDriveM.Timer1+lWaitTime;
            lState:=2;
        END_IF;
END_CASE;
```

```
Global Variables 0
Globale_Variablen
Globale_Variablen
Implementation of POU BetriebsanzeigeH1
Error: Project must contain a POU named PLC_PRG (main routine)
Code Size: 4572, Data Size: 236948
1 Error(s).
```

Fig. 5-25: Enter user program

An error message appears!
This error message indicates that no program has been entered in the task configuration.

➢ Select Resource to change to the task configuration.

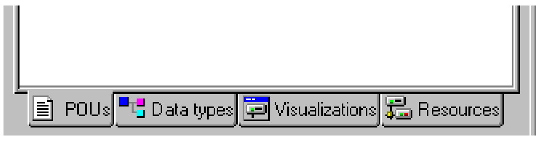


Fig. 5-26: Selection of resources in the Object Organizer

➢ Select *Task configuration* to reach the corresponding editor.

PD_UserMan_Richtlinien_us.fm

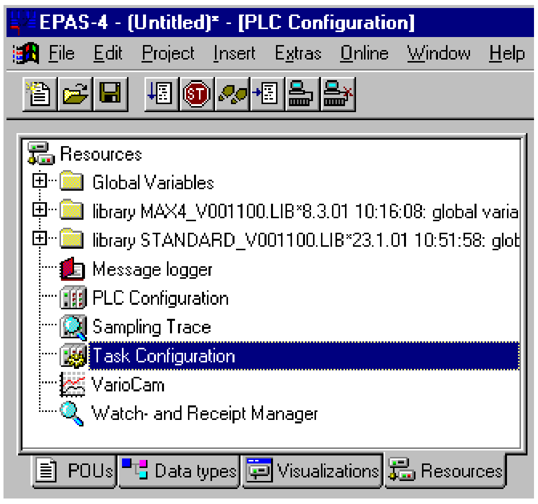

EPAS-4 - (Untitled)* - [PLC Configuration]
File Edit Project Insert Extras Online Window Help
Resources
Global Variables
library MAX4_V001100.LIB*8.3.01 10:16:08: global varia
library STANDARD_V001100.LIB*23.1.01 10:51:58: glob
Message logger
PLC Configuration
Sampling Trace
Task Configuration
VarioCam
Watch- and Receipt Manager
POUs
Data types
Visualizations
Resources

Fig. 5-27: Select task configuration

First you have to create a task now.

➢ Click with the right mouse button and select "Add task".

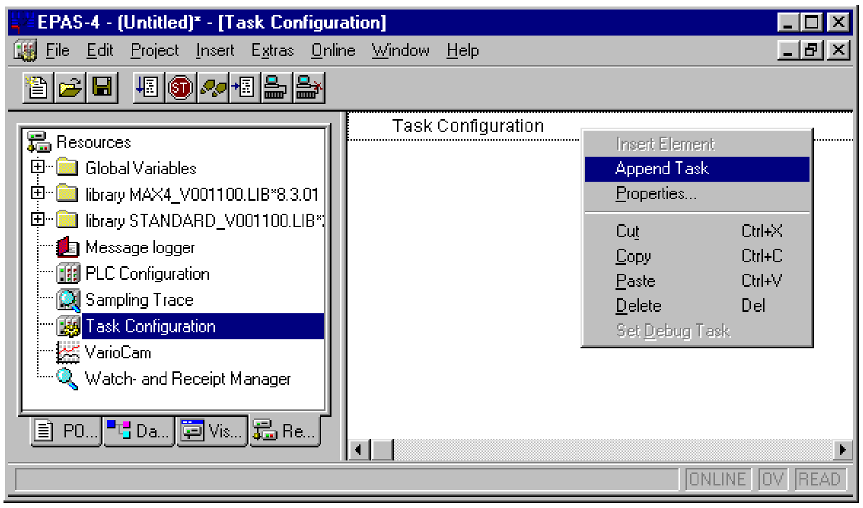
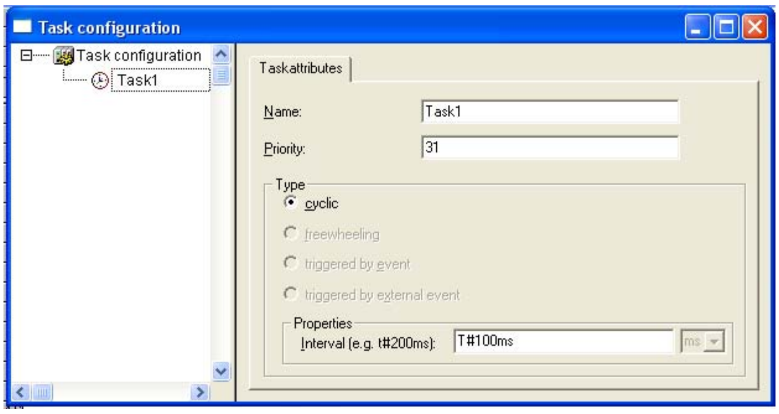


Fig. 5-28: Add task

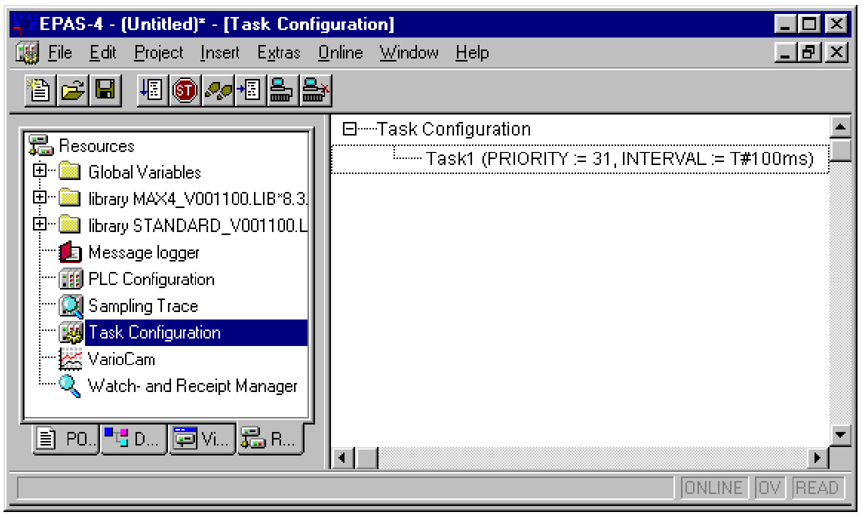
➢ Now give the task a name in the dialog box. For the time being, leave the priority at 31. As interval, please enter 100 ms.



Name: Task1
Priority: 31
Type: [✓ cyclic] freewheeling, triggered by event, triggered by external event
Interval (e.g. t#200ms): T#100ms

Fig. 5-29: Define task properties

- Confirm your entry with “OK“.


EPAS-4 - [Untitled]* - [Task Configuration]
File Edit Project Insert Extras Online Window Help
Resources
Global Variables
library MAX4_V001100.LIB*8.3.
library STANDARD_V001100.L
Message logger
PLC Configuration
Sampling Trace
Task Configuration
VarioCam
Watch- and Receipt Manager
PO.. D... Vi... R...
ONLINE OV READ

```
Task Configuration
    Task1 (PRIORITY := 31, INTERVAL := T#100ms)
```

Fig. 5-30: Task configuration with the new task

Now you have to add the newly created program to the task. Proceed as follows:

- Click with right mouse button on Task1 (text in brackets)
- Select "Append Program Call"

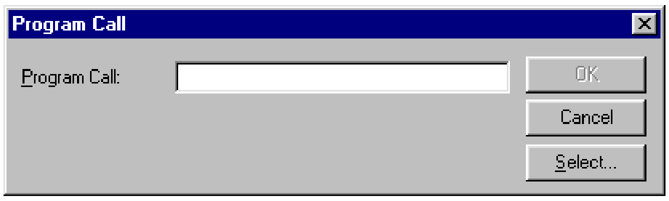


Fig. 5-31: Dialog box: Program call

PD_UserMan_Richtlinien_us.fm

- Go Select to open a selection of existing programs.

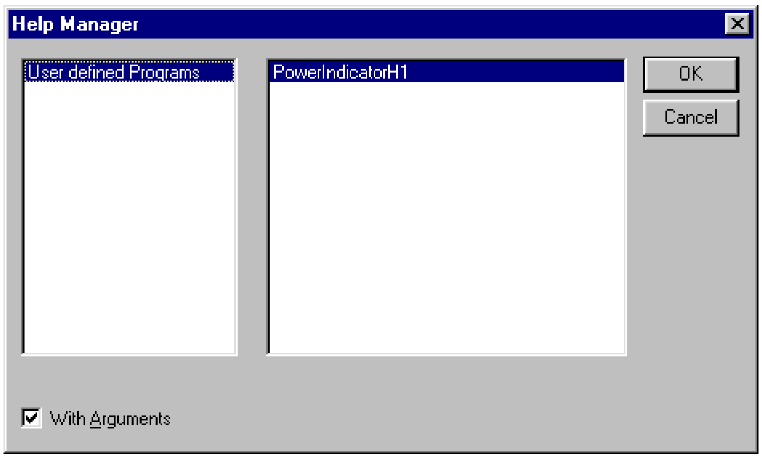
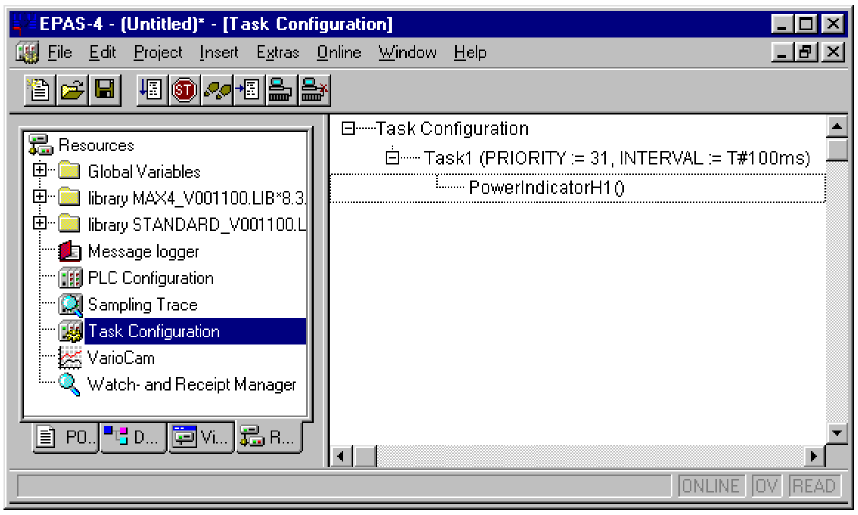


Fig. 5-32: Assistant for dialog “Program call“

- Select the program “PowerIndicatorH1“ and confirm with OK
- Confirm selection in dialog “Program call“ with OK


EPAS-4 - (Untitled)* - [Task Configuration]
File Edit Project Insert Extras Online Window Help
Resources
Global Variables
library MAX4_V001100.LIB*8.3.
library STANDARD_V001100.L
Message logger
PLC Configuration
Sampling Trace
Task Configuration
VarioCam
Watch- and Receipt Manager
PO... D... Vi... R...
ONLINE OV READ

Task Configuration
Task1 (PRIORITY := 31, INTERVAL := T#100ms)
PowerIndicatorH1()

Fig. 5-33: Task configuration with new task and program call

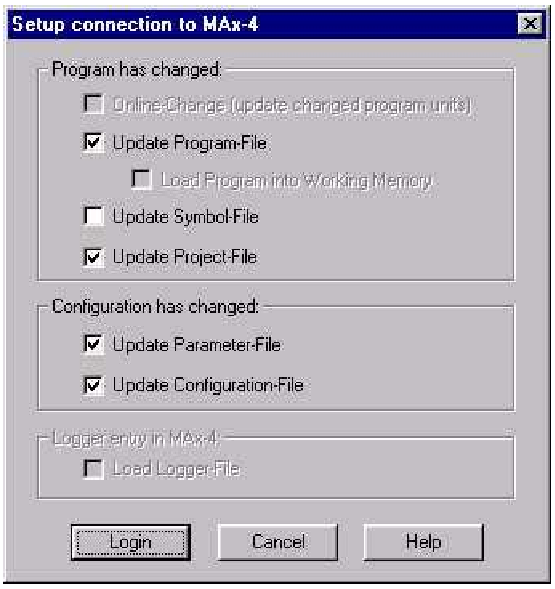
- Now check the program again, using “Compile all" (F11).

No errors have occurred; the project can be transmitted to the PacDrive Controller.

5.7.5 Observe & Operate

Transmit a project

- To transmit the program to the PacDrive Controller, select “Login“ from the menu item ONLINE in EPAS-4.
  You can also establish a connection with the PacDrive Controller by using the symbol *Login*.
- Answer the following question by selecting “Login".



Setup connection to MAx-4

Program has changed:
- [ ] Online-Change (update changed program units)
- [x] Update Program-File
  - [ ] Load Program into Working Memory
- [ ] Update Symbol-File
- [x] Update Project-File

Configuration has changed:
- [x] Update Parameter-File
- [x] Update Configuration-File

Logger entry in MAx-4:
- [ ] Load Logger-File

Fig. 5-34: Login dialog box

Now the program will be transmitted to the PacDrive Controller. As the configuration has changed, the PacDrive Controller must be restarted!

Test a program

The following functions can be executed under the menu item ONLINE:

- Start
  The program is restarted or continued with the next command after a stop.
- Stop
  The IEC program is interrupted.
- Reset
  The IEC program is terminated.

You can also set breakpoints an then e.g. go through a program task in single step.

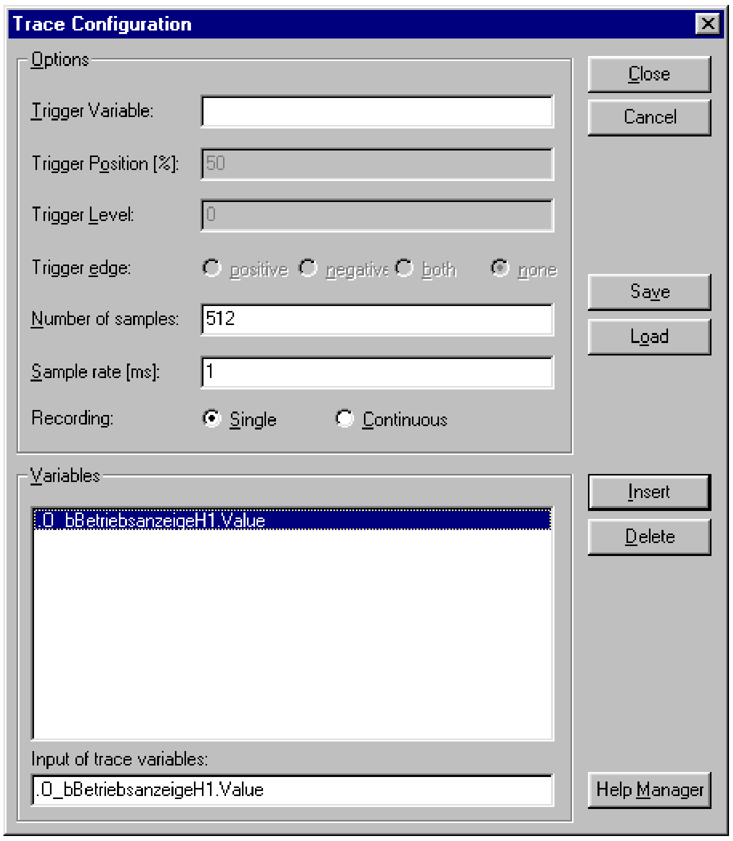
Watch and receipt administrator

You can enter all relevant object parameters and variables in a list in the watch and receipt administrator and observe them in ONLINE operation without having to scroll up and down or opening several windows.

Trace recording

Trace recording helps to record object parameter values and variable values chronologically. This function is a kind of oscilloscope.

To be able to create a trace record, you first need to configure the trace.


Trace Configuration

Options

Trigger Variable:

Trigger Position [%]: 50

Trigger Level: 0

Trigger edge: positive negative both [✓ none]

Number of samples: 512

Sample rate [ms]: 1

Recording: [✓ Single] Continuous

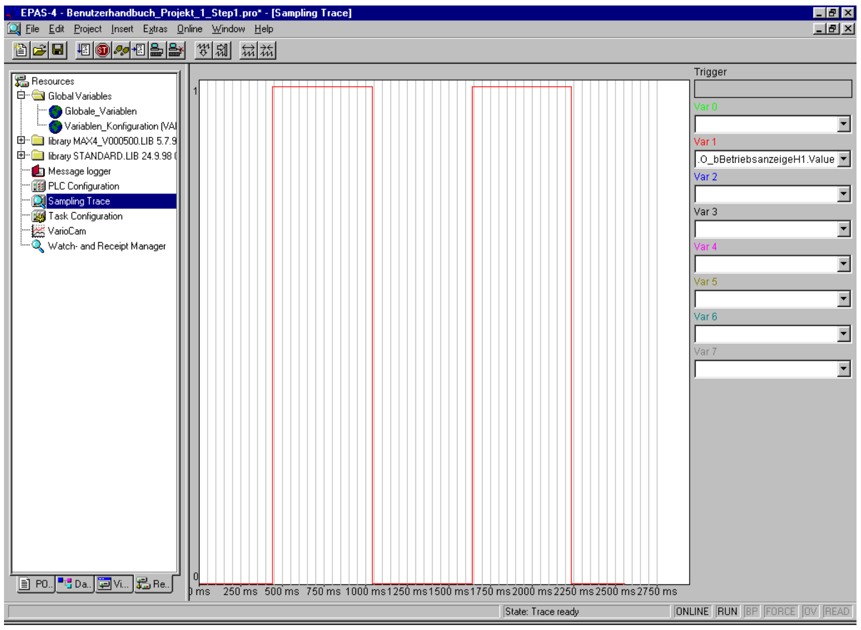
Variables

.0_bBetriebsanzeigeH1.Value

Input of trace variables: .0_bBetriebsanzeigeH1.Value

Fig. 5-35: Trace configuration

A record of your operating display looks for example like this:


EPAS-4 - Benutzerhandbuch_Projekt_1_Step1.pro* - [Sampling Trace]
File Edit Project Insert Extras Online Window Help
Resources
Global Variables
Globale_Variablen
Variablen_Konfiguration (VAI
library MAX4_V000500.LIB 5.7.9
library STANDARD.LIB 24.9.98 (
Message logger
PLC Configuration
Sampling Trace
Task Configuration
VarioCam
Watch- and Receipt Manager
PO.. Da.. Vi... Re..
1
0
0 ms 250 ms 500 ms 750 ms 1000 ms 1250 ms 1500 ms 1750 ms 2000 ms 2250 ms 2500 ms 2750 ms
Trigger
Var 0
Var 1
Var 2
Var 3
Var 4
Var 5
Var 6
Var 7
ONLINE RUN BP FORCE OV READ

Var 1: .O_bBetriebsanzeigeH1.Value

State: Trace ready

Fig. 5-36: Trace recording of equipment-on indicator H1

NOTE

For further explanations on the trace function, see Operating Manual EPAS-4.

If you checked your record thoroughly, you certainly noticed that the frequency at which our equipment-on indicator is flashing is not exactly 1 Hz.

Why is that?

The IEC program we wrote is only called up every 100 ms. You set this in the task configuration.
Due to this setting, the “waiting time“ of 500 ms can fluctuate between 501 and 600 ms.

For our equipment-on indicator, however, this does not matter since only a lamp is addressed.

5.7.6 Document and Save a Project

The complete documentation of your project includes the following parts:

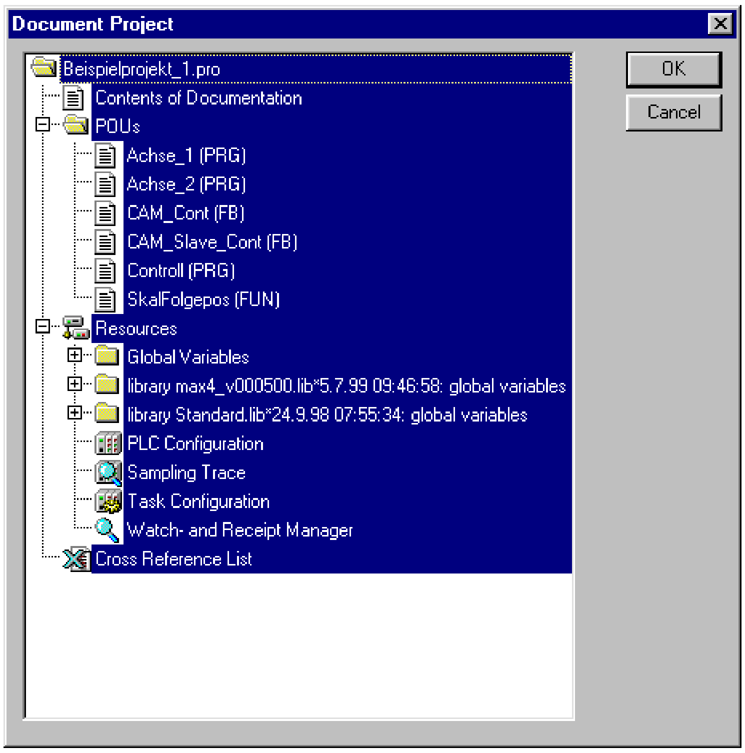
- performance specification
- EPAS-4 project file and maybe further supplementary files
- software and hardware versions of the components

- test reports
- acceptance protocol
- documentation print-out from EPAS-4

Make surey you use "talking" variable names and comment the program components. Otherwise it will be much harder to service the project.

Program changes should always be documented and also identfiable from the project name. This can be achieved by means of a vesion number or by including the date in the name of the project file.

In EPAS-4, there is the menu item PROJECT | DOCUMENTATION in order to support the user in his documenation.



```
Beispielprojekt_1.pro
  Contents of Documentation
  POUs
    Achse_1 (PRG)
    Achse_2 (PRG)
    CAM_Cont (FB)
    CAM_Slave_Cont (FB)
    Controll (PRG)
    SkalFolgepos (FUN)
  Resources
    Global Variables
    library max4_v000500.lib*5.7.99 09:46:58: global variables
    library Standard.lib*24.9.98 07:55:34: global variables
    PLC Configuration
    Sampling Trace
    Task Configuration
    Watch- and Receipt Manager
  Cross Reference List
```

Fig. 5-37: Dialog “Project documentation“ in EPAS-4

5.8 Expansion of the Example by Positioning Functions

5.8.1 Problem Definition

Now we want to expand the “simple“ project by positioning functions on a motor. Here is the extended problem definition:

An equipment-on indicator H1 is to flash on the unit as soon as the unit is active. Packaging material is to be delivered on a belt to the cutting facility as soon as a start switch has been activated. If the start switch is switched off again, the plant is to stop.

Describe the motion profile

The motion profile of the conveyer belt consists of a start cam, which is driven once when the start switch is activated, a straight line, which runs permanently during operation, and a stop cam, which is driven when the plant is switched off (start switch OFF).

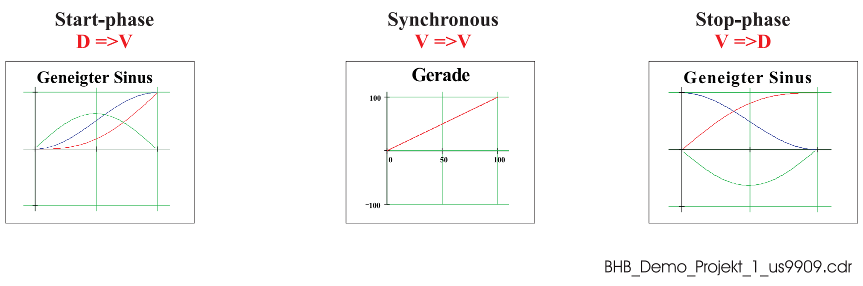


Fig. 5-38: Motion profile

Define input and ouput values, sensors and actuators

Opera-ting means	Input/output designation	Comment
H1	O_bEquipmentOnIndicatorH1	An indicator lamp is connected to the output. It will flash with a frequency of 1 Hz as soon as the unit is switched on.
S1	I_bStartS1	On -> start unit or unit is active Off -> stop unit or unit is stopped

Table 5-9: I/O definition

Depict the development over the time

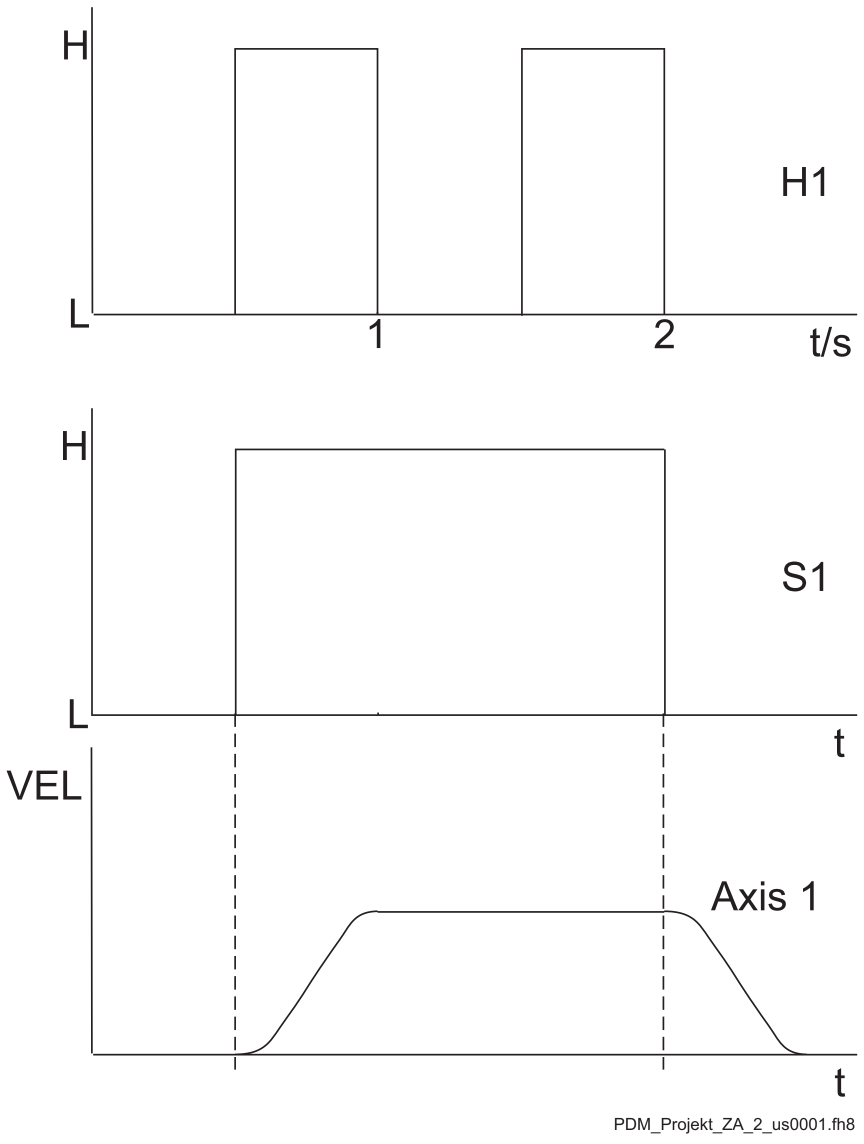


Fig. 5-39: Development over the time

5.8.2 Hardware

To be able to solve the problem, a PacDrive MC-4 is now needed in addition to the MAx-4 PacDrive Controller.

5.8.3 Configuration

Configure the project

To be able to realize the project, an additional MC-4 MotorControl-ler with SM motor is needed to drive the conveyer belt.

The MC-4 MotorController is entered in the control configuration as follows:

- Click with the right mouse button on Real-time bus

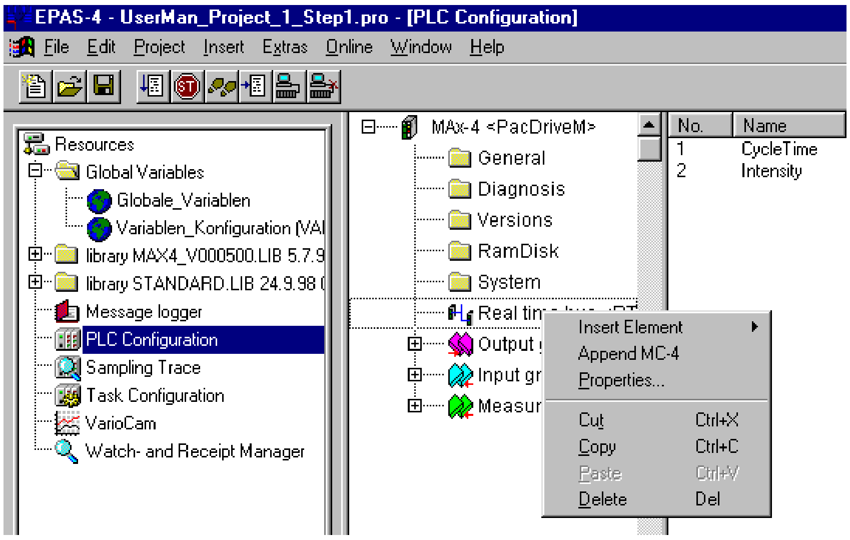


Fig. 5-40: Insert MC-4 MotorController in control configuration

- Select “Add MC-4“
- Assign the "symbolc" name "conveyer_belt" to the MC-4.

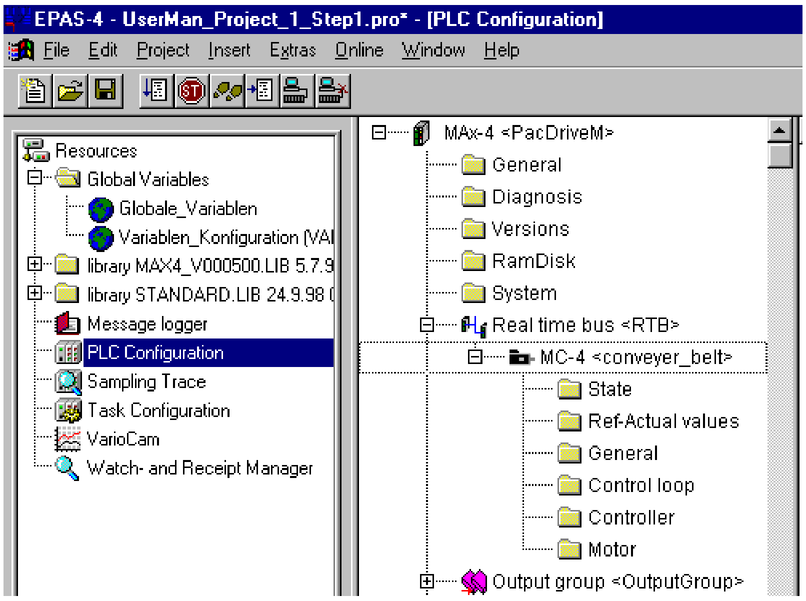


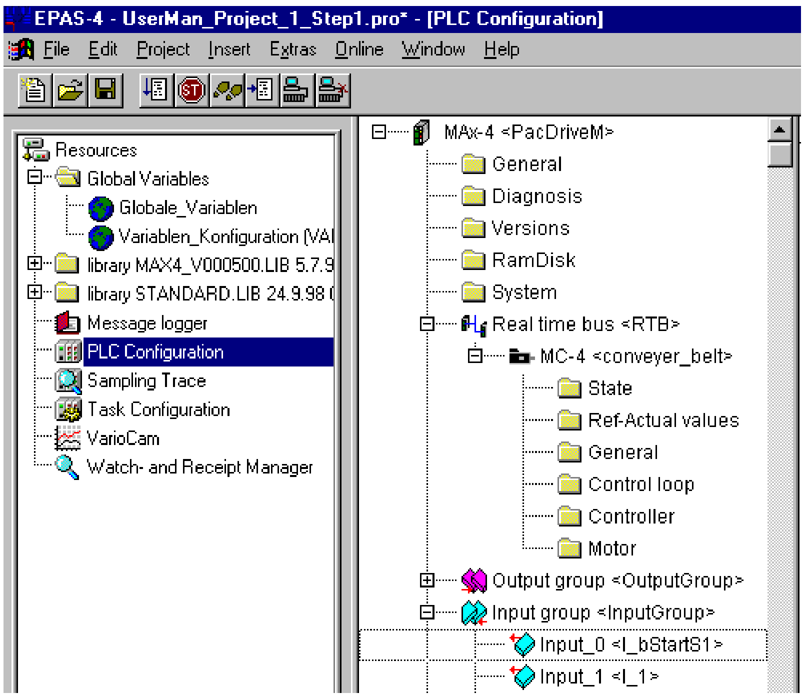
Fig. 5-41: Assign a symbolic name to MC-4 MotorController

NOTE

As we want to “simulate“ the MC-4 MotorController with the motor for the time being, no further settings in the control configuration are required at this stage.

Next we want to give the input S1 a symbolic name. We already defined this name in the problem definition.

➢ Enter the name in the control configuration.


EPAS-4 - UserMan_Project_1_Step1.pro* - [PLC Configuration]

File Edit Project Insert Extras Online Window Help

Resources
- Global Variables
  - Globale_Variablen
  - Variablen_Konfiguration (VAI
- library MAX4_V000500.LIB 5.7.9
- library STANDARD.LIB 24.9.98 (
- Message logger
- PLC Configuration
- Sampling Trace
- Task Configuration
- VarioCam
- Watch- and Receipt Manager

MAx-4 <PacDriveM>
- General
- Diagnosis
- Versions
- RamDisk
- System
- Real time bus <RTB>
  - MC-4 <conveyer_belt>
    - State
    - Ref-Actual values
    - General
    - Control loop
    - Controller
    - Motor
- Output group <OutputGroup>
- Input group <InputGroup>
  - Input_0 <I_bStartS1>
  - Input_1 <I_1>

Fig. 5-42: Create a symbolic name for Input_0

As the motion profile consists of cams, a master encoder is needed.

NOTE

ELAU calls the principle of cam treatment and processing Vario-Cam™. An axis (following axis) follows a master position (master axis). The cam sets the following position in dependence of the master position.
To forward master positions in the IEC program to cam functions, a logic encoder is needed.

PD_UserMan_Richtlinien_us.fm

The logic encoder is entered as follows:

- Click with the right mouse button on MAx-4

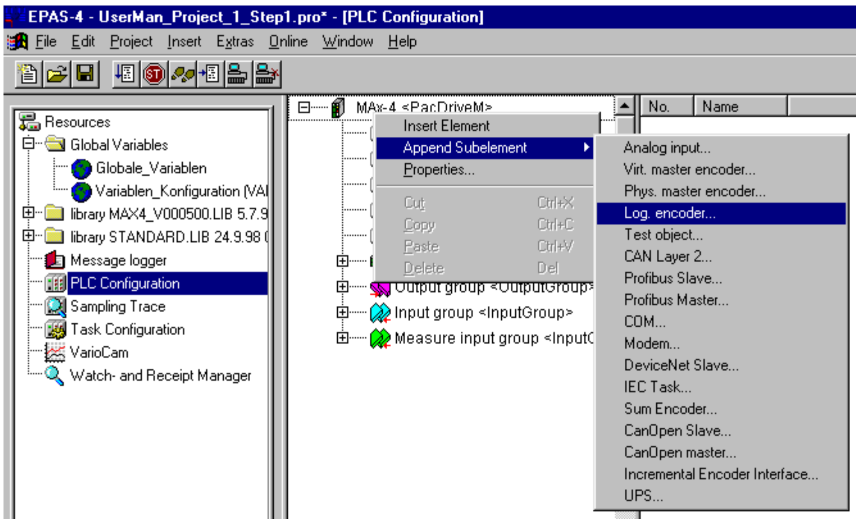


Fig. 5-43: Add a logic encoder to the configuration

- Select logic encoder
- Assign the symbolic name "MasterEncoder" to the logic encoder.

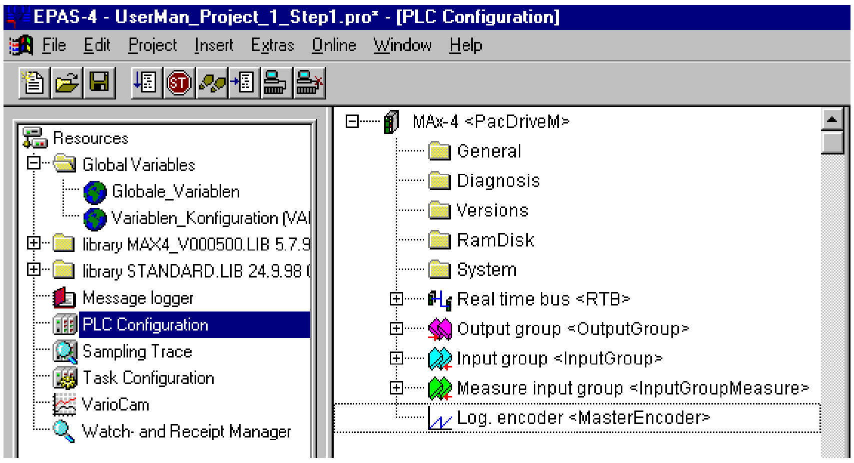

EPAS-4 - UserMan_Project_1_Step1.pro* - [PLC Configuration]
File Edit Project Insert Extras Online Window Help
Resources
Global Variables
Globale_Variablen
Variablen_Konfiguration (VAI
library MAX4_V000500.LIB 5.7.9
library STANDARD.LIB 24.9.98
Message logger
PLC Configuration
Sampling Trace
Task Configuration
VarioCam
Watch- and Receipt Manager
MAx-4 <PacDriveM>
General
Diagnosis
Versions
RamDisk
System
Real time bus <RTB>
Output group <OutputGroup>
Input group <InputGroup>
Measure input group <InputGroupMeasure>
Log. encoder <MasterEncoder>

Fig. 5-44: Assign symbolic name for logic encoder

5.8.4 User Program

Now create a new program (as described above) to program the motion function for the conveyer belt.

- First you have to load the cam profiles and adapt them to the problem:

```
(* load profiles*)
StartKurveId:=ProfilLoad('modisin');
ProfilSetLambda(StartCamId, 1.0);
(* -> dwell -> velocity *)

CyclicCamId:=ProfilLoad('straight');
(* -> velocity -> velocity *)

StopCamId:=ProfilLoad('modisin');
ProfilSetLambda(StopCamId, 0.0);
(* -> velocity -> dwell *)
```

NOTE

The ELAU libraries are described in detail in the EPAS-4 Online help.

- Now the individual cams have to be scaled:

```
(* scale axes---------- *)
lrYFactor1:=ScalFollPos(1.0, g_lrPartLength,
g_K_MS);(* start cam *)
lrYFactor2:=ScalFollPos(1.0, g_lrPartLength,
g_K_ST);(* cyclic cam *)
lrYFactor3:=ScalFollPos(1.0, g_lrPartLength,
g_K_MS);(* stop cam *)
```

NOTE

The function “ScalFollPos“ takes over the calculation for *lrYFactor*. It merely makes the following calculation:
ScalFollPos:=(m * sMasterPos) / K;

- Now the actual positioning procedure is programmed:

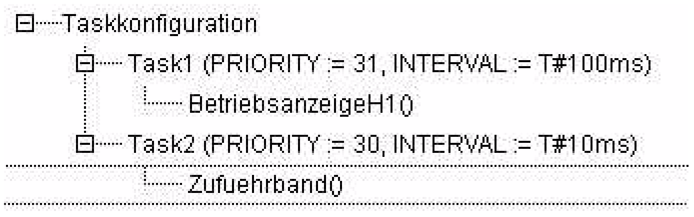
```
(* start cam ---------------------------------- *)
4:(* prepare start cam*)
  bCamReset:=TRUE;
  g_lState_1:=g_lState_1+1;
5:(* start start cam*)
  CAM(lAxisId:= _ConveyerBelt, lEncId:=
_MasterEncoder,
  lProfilId:= StartCamId, lrXOffset:= 0.0,
  lrYOffset:= 0.0, lrXFactor:= g_lrPartLength,
  lrYFactor:= lrYFactor1,lrXLimMin:= 0.0,
  lrXLimMax:= g_lrPartLength,bXLimMinOn:= TRUE,
  bXLimMaxOn:= TRUE, iXSetposMode:= ABSOLUTE,
    lrXSetposPos:= 0.0, iYSetposMode:= ABSOLUTE,
    lrYSetposPos:= 0.0, bReset:= bCamReset);
    bCamReset:=FALSE;
    IF (CAM.lResult = 0) THEN
       Schlz_DC:=0;
       bCamReset:=TRUE;
       g_lState_1:=g_lState_1+1;
    END_IF;
 (* cyclic cam ----------------------------- *)
 (* 1st cycle must have different position manupula-
 tion! *)
 6:(* start cyclic cam *)

    :
```

➢ Now you have to program the positioning coordinates.

NOTE

The complete example project can be found on the EPAS-4 CD.

➢ Create a new *task* in the *task configuration* and enter the program “conveyer_belt“. Please note that the priority of the program “conveyer_belt“ must be higher than that of the equipment-on indicator. Set priority = 30. As interval time, select 10 ms, so that the cams can be linked fast enough and there will be no "jumps" in the positioning movement.



```
⊟Taskkonfiguration
   ⊟Task1 (PRIORITY := 31, INTERVAL := T#100ms)
      BetriebsanzeigeH1()
   ⊟Task2 (PRIORITY := 30, INTERVAL := T#10ms)
      Zufuehrband()
```

Fig. 5-45: Task configuration with the two tasks of different priorities

5.8.5 Observe & Operate

Transmit the project

Now log into the PacDrive Controller (as described above).

Program test / debugging

In addition to the possibilities presented in the first version of our project, there are further possibilities, which are desribed below.

<u>Single step</u>

The program can be run in single-step mode. Proceed as follows:

- First you have to define the debug task. Go to the *task configuration* and click with the right mouse button on the *task* you want to debug.

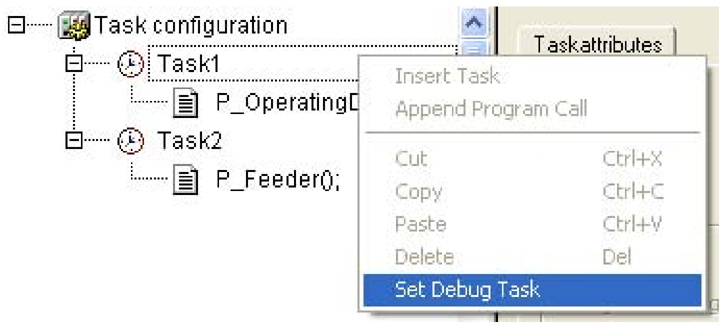


Fig. 5-46: Define debug task in the task configuration

- Select “Set Debug Task“. [DEBUG] appears as a marker behind the task.

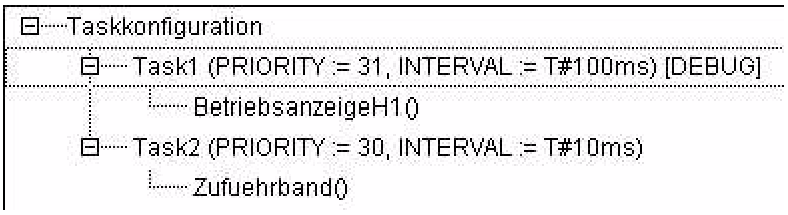

Taskkonfiguration
Task1 (PRIORITY := 31, INTERVAL := T#100ms) [DEBUG]
BetriebsanzeigeH1()
Task2 (PRIORITY := 30, INTERVAL := T#10ms)
Zufuehrband()

Fig. 5-47: Display of the debug task in the task configuration

- Now you have to set a breakpoint. A breakpoint is a point in the program at which processing is halted. To set a breakpoint, go to a POU and click with the left mouse button on the line number at which the program is to be halted. The following safety check appears:

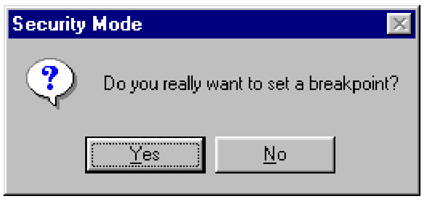
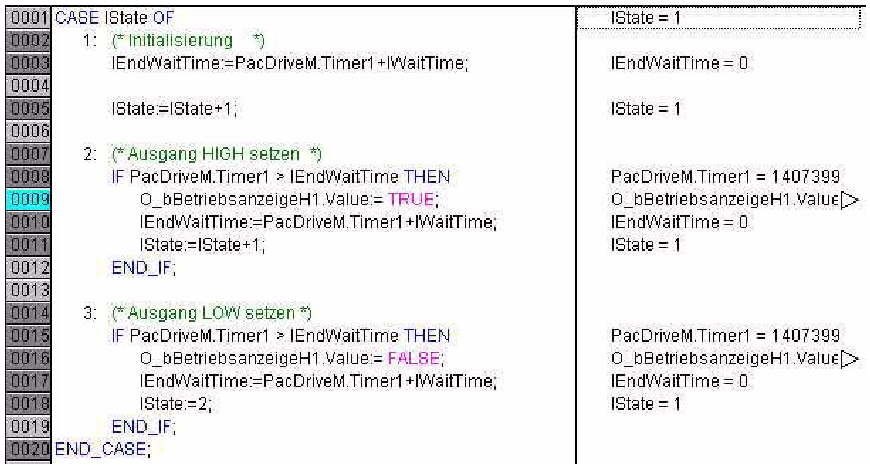


Fig. 5-48: Safety check when setting a breakpoint

- Confirm with “Yes“ to set the breakpoint. The line at which you set the breakpoint is marked in color.

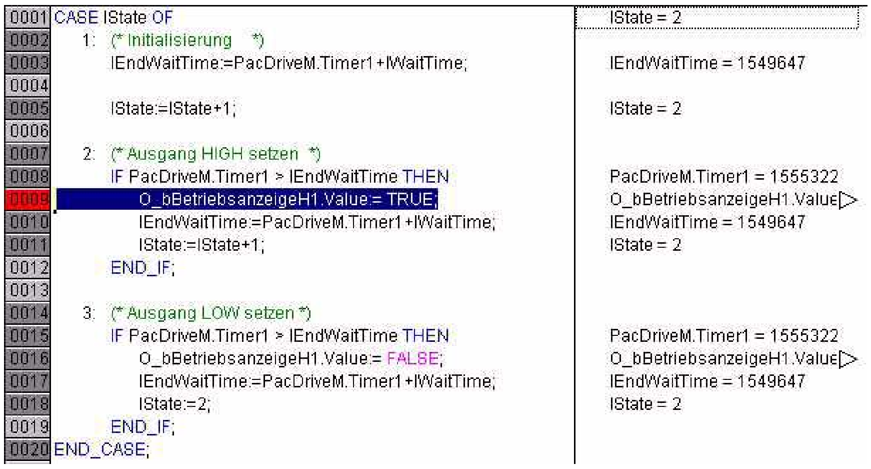
PD_UserMan_Richtlinien_us.fm



```
CASE lState OF                                        lState = 1
    1:  (* Initialisierung    *)
        lEndWaitTime:=PacDriveM.Timer1+lWaitTime;     lEndWaitTime = 0

        lState:=lState+1;                             lState = 1

    2:  (* Ausgang HIGH setzen  *)
        IF PacDriveM.Timer1 > lEndWaitTime THEN       PacDriveM.Timer1 = 1407399
            O_bBetriebsanzeigeH1.Value:= TRUE;        O_bBetriebsanzeigeH1.Value▷
            lEndWaitTime:=PacDriveM.Timer1+lWaitTime; lEndWaitTime = 0
            lState:=lState+1;                         lState = 1
        END_IF;

    3:  (* Ausgang LOW setzen *)
        IF PacDriveM.Timer1 > lEndWaitTime THEN       PacDriveM.Timer1 = 1407399
            O_bBetriebsanzeigeH1.Value:= FALSE;       O_bBetriebsanzeigeH1.Value▷
            lEndWaitTime:=PacDriveM.Timer1+lWaitTime; lEndWaitTime = 0
            lState:=2;                                lState = 1
        END_IF;
END_CASE;
```

Fig. 5-49: ST program editor in online operation with breakpoint

➢ Now start the program. As soon as the breakpoint is reached, the color mark will change.



```
CASE lState OF                                        lState = 2
    1:  (* Initialisierung    *)
        lEndWaitTime:=PacDriveM.Timer1+lWaitTime;     lEndWaitTime = 1549647

        lState:=lState+1;                             lState = 2

    2:  (* Ausgang HIGH setzen  *)
        IF PacDriveM.Timer1 > lEndWaitTime THEN       PacDriveM.Timer1 = 1555322
            O_bBetriebsanzeigeH1.Value:= TRUE;        O_bBetriebsanzeigeH1.Value▷
            lEndWaitTime:=PacDriveM.Timer1+lWaitTime; lEndWaitTime = 1549647
            lState:=lState+1;                         lState = 2
        END_IF;

    3:  (* Ausgang LOW setzen *)
        IF PacDriveM.Timer1 > lEndWaitTime THEN       PacDriveM.Timer1 = 1555322
            O_bBetriebsanzeigeH1.Value:= FALSE;       O_bBetriebsanzeigeH1.Value▷
            lEndWaitTime:=PacDriveM.Timer1+lWaitTime; lEndWaitTime = 1549647
            lState:=2;                                lState = 2
        END_IF;
END_CASE;
```

Fig. 5-50: Breakpoint reached

➢ Now you can continue running the program (or the POU) in *single-step*. You can carry out single steps using the symbol or the menu item ONLINE or the function keys.

PD_UserMan_Richtlinien_us.fm

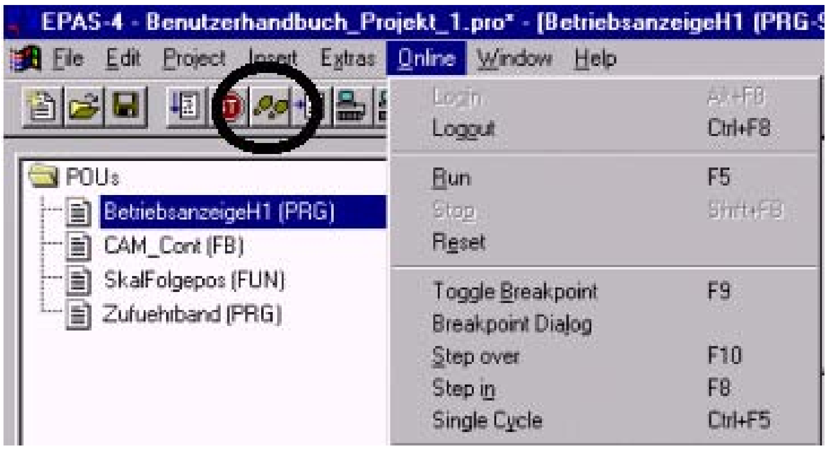


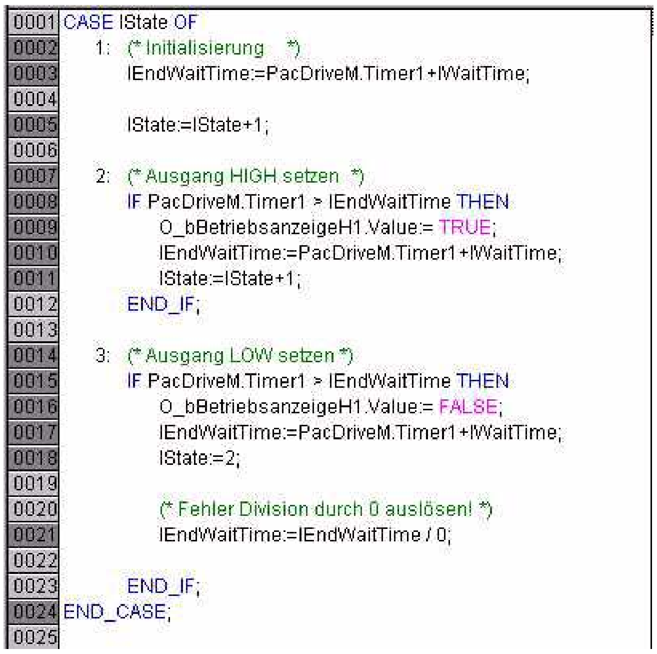
Fig. 5-51: Symbol for single-step and menu items Single-step

Call hierarchy

This function can be used to display the call stack in case of an actual diagnosis message.

Example:

➢ We will cause an error on purpose by extending the program as follows:

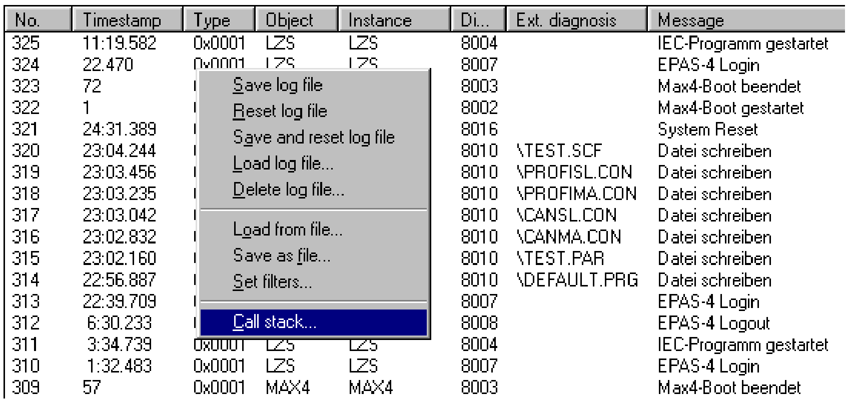


```
CASE lState OF
    1:  (* Initialisierung   *)
        lEndWaitTime:=PacDriveM.Timer1+lWaitTime;

        lState:=lState+1;

    2:  (* Ausgang HIGH setzen  *)
        IF PacDriveM.Timer1 > lEndWaitTime THEN
            O_bBetriebsanzeigeH1.Value:= TRUE;
            lEndWaitTime:=PacDriveM.Timer1+lWaitTime;
            lState:=lState+1;
        END_IF;

    3:  (* Ausgang LOW setzen *)
        IF PacDriveM.Timer1 > lEndWaitTime THEN
            O_bBetriebsanzeigeH1.Value:= FALSE;
            lEndWaitTime:=PacDriveM.Timer1+lWaitTime;
            lState:=2;

            (* Fehler Division durch 0 auslösen! *)
            lEndWaitTime:=lEndWaitTime / 0;

        END_IF;
END_CASE;
```

Fig. 5-52: ST program with error

➢ Start the program and wait until the diagnosis message is triggered (err-LED on MAx-4 flashing).

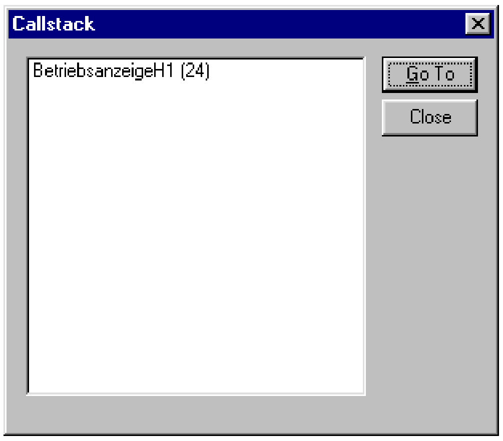
➢ Open the message logger and click with the right mouse button on the error message Division error.



No.	Timestamp	Type	Object	Instance	Di...	Ext. diagnosis	Message
325	11:19.582	0x0001	LZS	LZS	8004		IEC-Programm gestartet
324	22.470	0x0001	LZS	LZS	8007		EPAS-4 Login
323	72				8003		Max4-Boot beendet
322	1				8002		Max4-Boot gestartet
321	24:31.389				8016		System Reset
320	23:04.244				8010	\TEST.SCF	Datei schreiben
319	23:03.456				8010	\PROFISL.CON	Datei schreiben
318	23:03.235				8010	\PROFIMA.CON	Datei schreiben
317	23:03.042				8010	\CANSL.CON	Datei schreiben
316	23:02.832				8010	\CANMA.CON	Datei schreiben
315	23:02.160				8010	\TEST.PAR	Datei schreiben
314	22:56.887				8010	\DEFAULT.PRG	Datei schreiben
313	22:39.709				8007		EPAS-4 Login
312	6:30.233				8008		EPAS-4 Logout
311	3:34.739	0x0001	LZS	LZS	8004		IEC-Programm gestartet
310	1:32.483	0x0001	LZS	LZS	8007		EPAS-4 Login
309	57	0x0001	MAX4	MAX4	8003		Max4-Boot beendet

Fig. 5-53: Select call hierarchy in the message logger

➢ Select Call hierarchy...



BetriebsanzeigeH1 (24)

Fig. 5-54: Dialog: Call hierarchy

The last component is always the component at which the processing is standing at the moment.

After selecting the component and clicking on the button Go to, the selected component is loaded into a window and the line or network where the processing is at the moment is shown.

5.8.6 Document and Save the Project

NOTE

Again you should not forget to document and save four project! See also „Document and Save a Project“, page 130.

6 Error Search Strategies

In this chapter we will try to make it easier for you to look delibera-tely for errors in your system.

6.1 Method

LEDs on the units	The LEDs permit first conclusions as to the cause of the error.	see “LEDs“

control configuration	With the help of object parameters, the error can be narrowed down further in the control configuration.	see “Control Configuration“

message logger	The message logger permits a detailed error analyis. Thanks to the error history, even following errors can be recognized easily. The message logger is the most important tool for error analysis in the PacDrive™ M system.	see “Message Logger“

Table 6-1: Method for commissioning

6.2 LEDs

From the LEDs, first conclusions as to the cause of an error are possible.

The LEDs are contained in the chapter “Configuration / Programming / Diagnosis“ of the corresponding device-specific operating manual.

Further documentation:

- MAx-4 Controller operating manual
- MC-4 Servo Drive operating manual
- ...

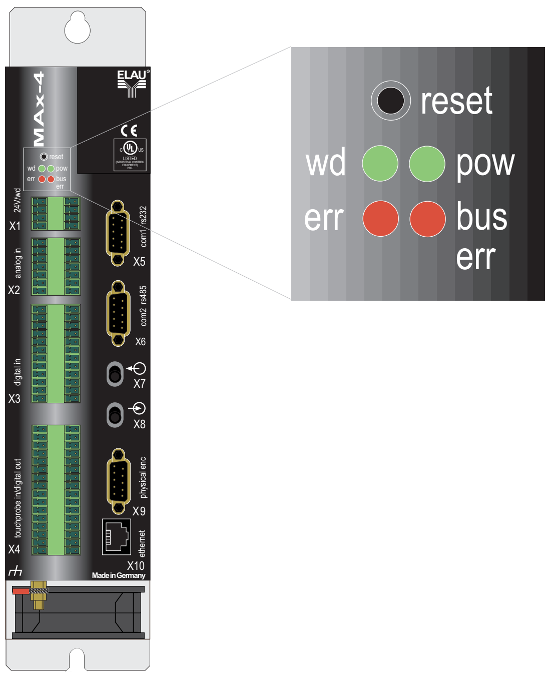
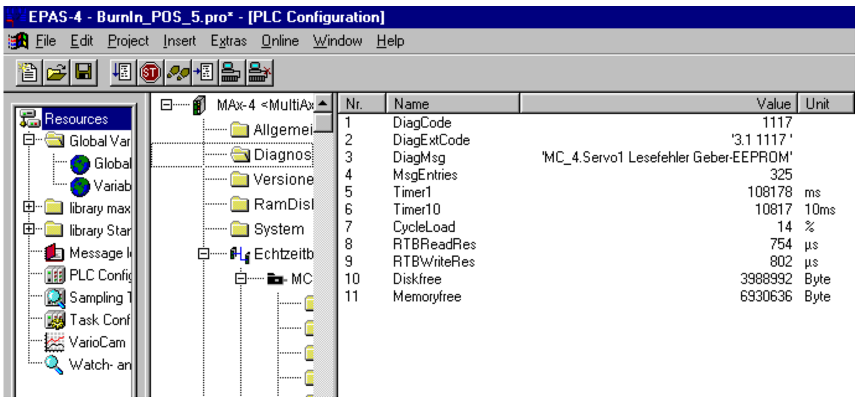


Fig. 6-1: Diagnosis LEDs of the MAx-4 PacDrive Controller

6.3 PLC configuration

For a more detailed diagnosis, you can use the diagnosis parameters in the control configuration.



Nr.	Name	Value	Unit
1	DiagCode	1117	
2	DiagExtCode	'3.1 1117 '	
3	DiagMsg	'MC_4.Servo1 Lesefehler Geber-EEPROM'	
4	MsgEntries	325	
5	Timer1	108178	ms
6	Timer10	10817	10ms
7	CycleLoad	14	%
8	RTBReadRes	754	µs
9	RTBWriteRes	802	µs
10	Diskfree	3988992	Byte
11	Memoryfree	6930636	Byte

Fig. 6-2: PLC configuration

The MAx-4 PacDrive Controller has the object parameters *DiagCode*, *DiagExtCode* and *DiagMsg*. The parameters contain the currently most important diagnosis information. They are only overwritten if a diagnosis message of a diagnosis class of higher priority occurs. Diagnosis messages of equal or lower priority will not change the parameter. Such diagnosis messages are only reset by quitting.

Example:

DiagCode	3117
DiagExtCode	"1.1 3117"
DiagMsg	"MC-4/axis1: motor temperature too high"

In case of axis-specific errors (sender: MC_4), the MC-4 MotorController object parameters *DiagCode* and *DiagMsg* are triggered. The contents of the parameters arise in analogy with the MAx-4 PacDrive Controller diagnosis parameters. The same rules apply for updating.

Example:

DiagCode	3117
DiagMsg	"motor temperature too high"

NOTE

For further information, please see EPAS-4 Online help, chapters “Control Configuration“ and “Diagnosis“.

6.4 Message Logger

6.4.1 Function

The message logger records system, diagnosis and usage messages in a message log buffer in the RAM range of the MAx-4 PacDrive Controller.

The errors are recorded in chronological order in a ring buffer (approx. 325 entries). A time stamp (ms since system start), the time, object, instance, diagnosis code, external diagnosis and message text are recorded.

As the message buffer in the RAM can only be buffered for a maximum of 7 days in case of power failure, it is possible to save the message log buffer in a messge log file on the disk in the MAx-4 PacDrive Controller. The MAx-4 PacDrive Controller can save several message log files on its disk.

The message log files can be transmitted from the MAx-4 PacDrive Controller to the PC with the help of EPAS-4. Several message log files can be saved there as well.

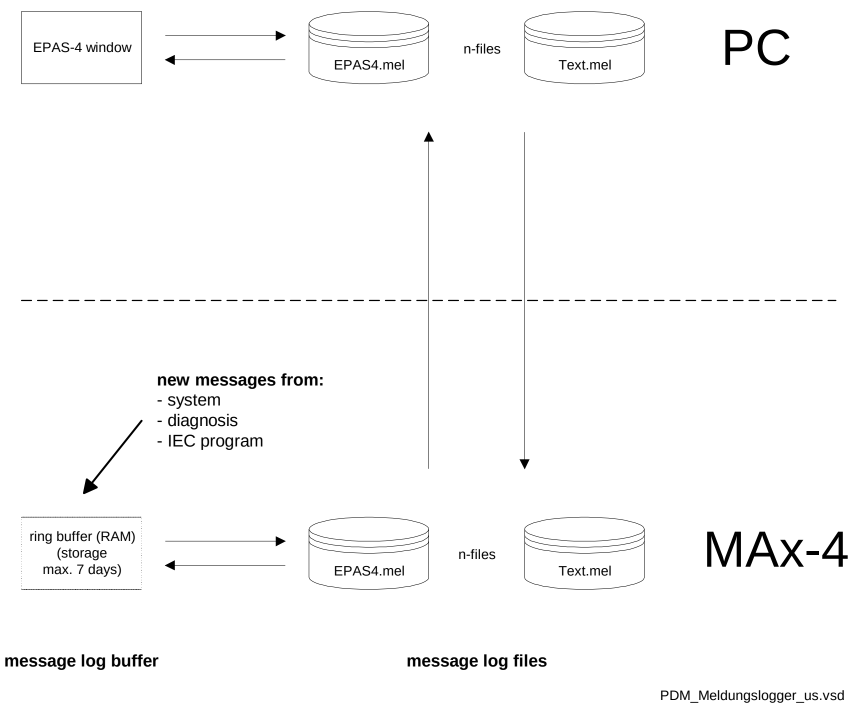


Fig. 6-3: Resources / Message logger

6.4.2 The Functions of a Message Logger in Detail

<u>Save and load a log file</u>

Load the current message logger from MAx-4 PacDrive Controller in EPAS-4.

<u>Save log file</u>

message log buffer (RAM)

-> message log file (MAx-4 file)

<u>Reset log entries</u>

clear message log buffer (RAM)

<u>Save and reset log file</u>

message log file (RAM)

-> message log file (MAx-4 file)

-> clear message log buffer (RAM)

<u>Load log file ...</u>

message log file (MAx-4 foile)

-> PC file

-> display

<u>Delete log file ...</u>

delete MAx-4 file

<u>Load from file ...</u>

PC file

-> display

<u>Save as file ...</u>

display

-> PC file

Functions	EPAS-4 / Message logger (menu item Tools)	IEC program
show current message logger in EPAS-4	save and load log file	
save log file	save log file	MsgLogSave()
reset log entries	reset log entries	MsgLogReset()
save and reset log file	save and reset log file	
load log file	load log file	
delete log file	delete log file	MsgLogDelete()
load from file ...	load from file ...	

Functions	EPAS-4 / Message logger (menu item Tools)	IEC program
save as file ...	save as file ...	
entry in message logger (RAM)		MsgLogInsert()

Table 6-2: Functions of the messge logger and access types

NOTE

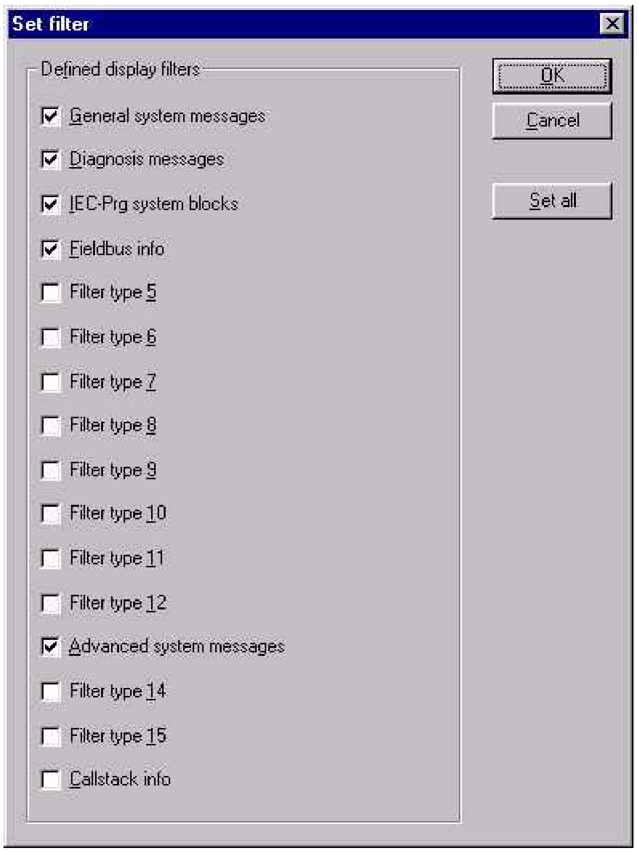
It is also possible to filter the message recording (see EPAS-4 Online help / Control Configuration / MAx-4 / MsgFilter) or to filter only the display of the messages (see next chapter).

6.4.3 Filtering Messages

The system messages in the message logger can be filtered in two ways:

- Filter when recording them from the system via the object parameter MsgFilter in the control configuration.
- Filter when displaying in EPAS-4 using the menu item TOOLS | SET FILTER...

The message logger differentiates between 16 different message classes. These classes can be switched on and off individualy.



Defined display filters

- [✓] General system messages
- [✓] Diagnosis messages
- [✓] IEC-Prg system blocks
- [✓] Fieldbus info
- [ ] Filter type 5
- [ ] Filter type 6
- [ ] Filter type 7
- [ ] Filter type 8
- [ ] Filter type 9
- [ ] Filter type 10
- [ ] Filter type 11
- [ ] Filter type 12
- [✓] Advanced system messages
- [ ] Filter type 14
- [ ] Filter type 15
- [ ] Callstack info

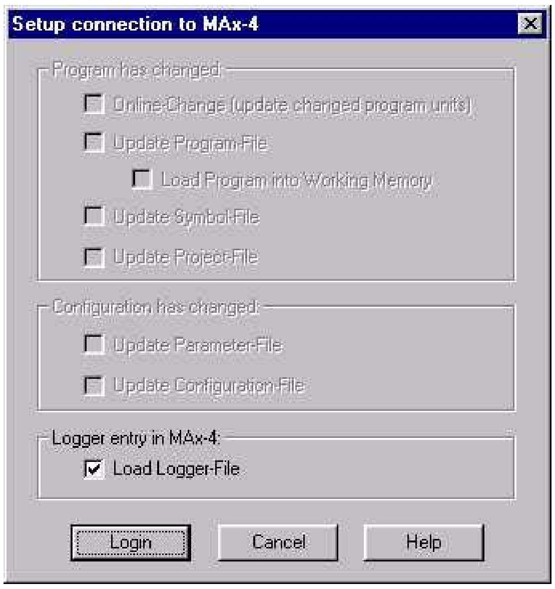
Fig. 6-4: Resources / Message logger / Dialog: Messge filter

NOTE

The individual classes are described in the EPAS-4 Online help under PLC Configuration.

6.4.4 Establish a Connection with the PacDrive Controller MAx-4 in Case of an Error

If you establish the connection with the MAx-4 PacDrive Controller in case of an error, the following selection box will appear:



Setup connection to MAx-4

Program has changed:

- [ ] Online-Change (update changed program units)
- [ ] Update Program-File
  - [ ] Load Program into Working Memory
- [ ] Update Symbol-File
- [ ] Update Project-File

Configuration has changed:

- [ ] Update Parameter-File
- [ ] Update Configuration-File

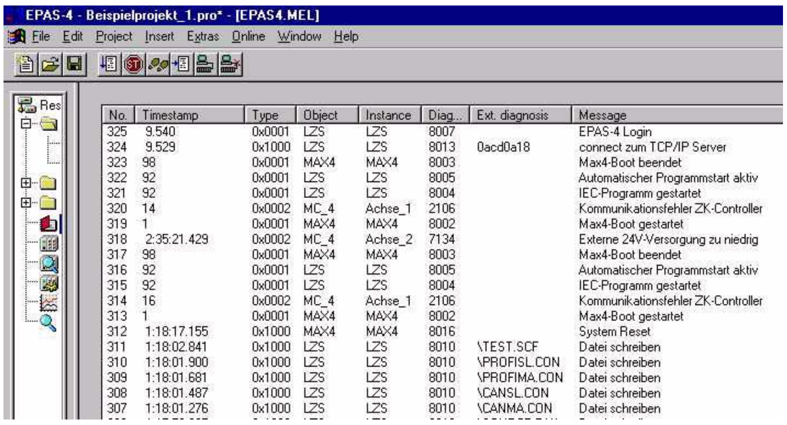
Logger entry in MAx-4:

- [✓] Load Logger-File

Fig. 6-5: Resources / Message logger / Establish a connection in case of an error

In case of an error, the menu item Load message file is selected.

➢ If you select LOGIN, the message file will be loaded from the MAx-4 PacDrive Controller in EPAS-4 and displayed.



EPAS-4 - Beispielprojekt_1.pro* - [EPAS4.MEL]

No.	Timestamp	Type	Object	Instance	Diag...	Ext. diagnosis	Message
325	9.540	0x0001	LZS	LZS	8007		EPAS-4 Login
324	9.529	0x1000	LZS	LZS	8013	0acd0a18	connect zum TCP/IP Server
323	98	0x0001	MAX4	MAX4	8003		Max4-Boot beendet
322	92	0x0001	LZS	LZS	8005		Automatischer Programmstart aktiv
321	92	0x0001	LZS	LZS	8004		IEC-Programm gestartet
320	14	0x0002	MC_4	Achse_1	2106		Kommunikationsfehler ZK-Controller
319	1	0x0001	MAX4	MAX4	8002		Max4-Boot gestartet
318	2:35:21.429	0x0002	MC_4	Achse_2	7134		Externe 24V-Versorgung zu niedrig
317	98	0x0001	MAX4	MAX4	8003		Max4-Boot beendet
316	92	0x0001	LZS	LZS	8005		Automatischer Programmstart aktiv
315	92	0x0001	LZS	LZS	8004		IEC-Programm gestartet
314	16	0x0002	MC_4	Achse_1	2106		Kommunikationsfehler ZK-Controller
313	1	0x0001	MAX4	MAX4	8002		Max4-Boot gestartet
312	1:18:17.155	0x1000	MAX4	MAX4	8016		System Reset
311	1:18:02.841	0x1000	LZS	LZS	8010	\TEST.SCF	Datei schreiben
310	1:18:01.900	0x1000	LZS	LZS	8010	\PROFISL.CON	Datei schreiben
309	1:18:01.681	0x1000	LZS	LZS	8010	\PROFIMA.CON	Datei schreiben
308	1:18:01.487	0x1000	LZS	LZS	8010	\CANSL.CON	Datei schreiben
307	1:18:01.276	0x1000	LZS	LZS	8010	\CANMA.CON	Datei schreiben

Fig. 6-6: Resources / Message logger / Message file

PD_UserMan_Fehlersuche_us.fm

Meaning of the entries in the message logger

No.

Consecutive number of the logger entries. The highest number is the latest logger entry.

Time stamp

Time since switching on the MAx-4 PacDrive Controller in the format:

DDD HH:MM:SS.sss

with

- DDD - days
- HH - hours (0 ... 23)
- MM - minutes (0 ... 59)
- SS - seconds (0 ... 59)
- sss - milliseconds (0 ... 999)

In versions < 00.07.00 the time is shown in ms.

Type

Shows the filter type of the message.

See also EPAS-4 Online help, Control Configuration MAx-4.General.MsgFilter.

Object

Shows the type of the object that caused the logger entry.

See also EPAS-4 Online help / Control Configuration.

Instance

Shows the instance of the object that caused the logger entry.

See also EPAS-4 Online help / Control Configuration.

Diag. Code

Shows the diagnosis code.

See also EPAS-4 Online help / Diagnosis.

Ext. diagnosis

Provides optional additional information on the diagnosis code. The information is also shown as optional additional information in the control configuration in MAx-4.diagnosis.DiagExtCode.

Message text

Shows the corresponding message text for the diagnosis code.

6.5 Further Possibilities for Diagnosis

NOTE

For further information, please see Operating Manuals, chapter “Diagnosis and Maintenance" and EPAS-4 Online help, chapter "Diagnosis".

6.5.1 Trace

The purpose of the trace recording is to record the time relations and the dynamic behavior of the system.

NOTE

For further information, see Operating Manual EPAS-4.

6.5.2 Debugging

The PacDrive™ system provides comfortable program debugging facilities for the user.

See also „Program test / debugging“, page 138.

6.5.3 Check Library

If the “Check“ library is integrated in the project, the function block Check, which checks the limits of array elements, is automatically called up every time an array element is accessed. Moreover, division errors (division by 0) are recognized.

NOTE

For the reason described above, the integration of the “Check“ library increases the runtime. The “Check“ library should therefore be removed from the project once the test phase is completed.

6.5.4 Call Hierarchy (CallStack)

You can start the call stack command (“Online > Call hierarchy“) when the project stops at a breakpoint. A list of components actually in the call stack is shown.

Example for a call hierarchy:

The first component is always PLC_PRG, as this is where the processing starts.

The last component is always the one that is up for processing at that particular moment.

After you selected one of the components and clicked on the button "go to", the selected component will be loaded into a window, and the line or network in which the processing is will be shown.

7 Teleservice

7.1 Overview

In recent years, the significance of teleservice in the capital goods industry has increased dramatically. This trend results not only from the **I**nformation **a**nd **C**ommunication technologies (IaC) available for teleservice (or remote service), but also from the possibility to create new unique selling points on the global markets. Teleservice means not only remote support for fault clearance. Rather, the use of teleservice technologies opens up new and above all innovative approaches for customer service over the entire life cylce of the packaging machine (“after-sales service“). This is why also terms like “tele-engineering“ or “teleconsulting“ are used.

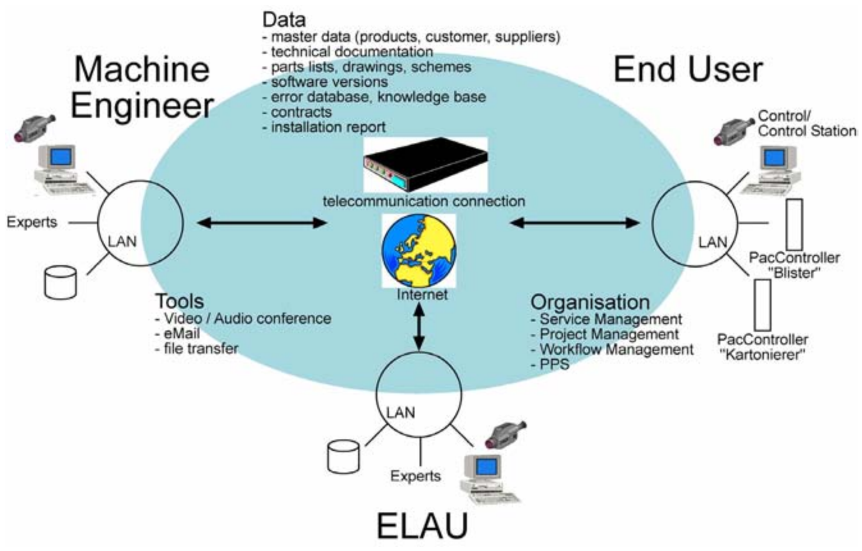


Fig. 7-1: Teleservice concept

Definition

Teleservice is defined as the support of customer service and application departments by IaC components and services, which make it possible to execute diagnosis and solve problems on machines from a remote place. Teleservice is used for the installation of new machines and plants, for fault clearance and for the transmission of new software versions. In the future, new applications for teleservice will include process support and customer consulting.

Connection of the PacDrive Controller

The MAx-4 PacDrive Controller (server) has a RS232 and an Ethernet interface. The RS232 interface is the basis for serial communication (connection of a zero modem cable or a modem). The Ethernet interface permits a direct connection of the control to a LAN (local area network) with TCP/IP support. The standard client (PC) also has a COM interface and an Ethernet card with appropriate connection. Thus, both sides have the physical interfaces to the outside world.

There are different options for connecting to the PacDrive™ System via a client:

- local Ethernet,
- direct telecommunication connection (via modem) and
- internet connection.

<u>Local Ethernet</u>

The first way to do a telediagnosis on the controller is a simple Ethernet connection. As the figure below shows, server and client must be connected via a TCP/IP network. The network can e.g. be installed within a company and connect several computers in a network.

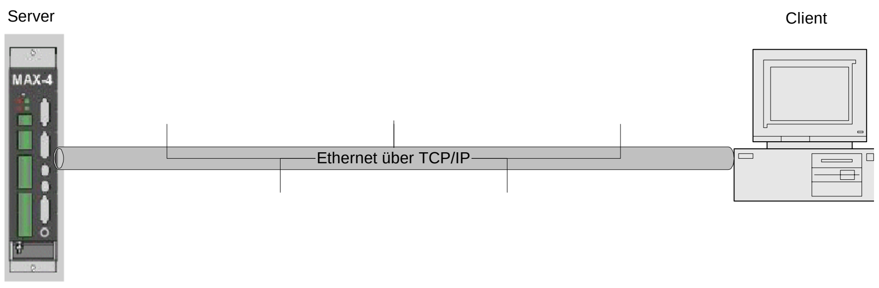


Fig. 7-2: Local Ethernet network

This method requires the TCP/IP protocol. Both computers must have an IP address, which must be unambiguous in the network.

Ethernet by default supports the connection via a local Ethernet. Therefore it will not be described here.

<u>Direct telecommuncation connection</u>

A direct modem connection to a control can be established with the help of the EPAS-4 Automation Toolkit.

Advantage:
Full access to all functions of the system.

Disadvantage:
You can only establish a direct connection between two computers. EPAS-4 must be installed on the client computer. This means that the system is no longer independent of platforms.

Internet connection

Teleservice is to ensure worldwide access to the MAx-4 PacDrive Controller. The World Wide Web is particularly suitable for this purpose. No matter where you are on the globe, you can dial up to the Internet with a PC, modem and telephone connection.

Advantage:
Remote diagnosis can be used independent of location and platform. EPAS-4 needs not be installed.

Disadvantage:
Very restricted system access.

7.2 Direct Telecommunication Connection

By all means observe the following information:

NOTE

As the modem is connected to the serial interface COM1 of the MAx-4 PacDrive Controller, a serial EPAS-4 connection is not possible at the same time. Therefore you should always have a functioning TCP/IP connection for EPAS-4 if you use telservice via modem connection.

7.2.1 Principle

In principle, teleservice via a direct telecommunication connection with a modem and EPAS-4 enables the user to use all the functions that are available in case of a “direct“ connection.

WARNING!

Restricted view of the machine!

Risk of injuries and material damage!

- Only do teleservice if a qualified person is available on site, i.e. at the machine.

- Always keep in touch with this person (e.g. via telephone or video conference).

The prerequisite for this is a modem connection (on client and server side).

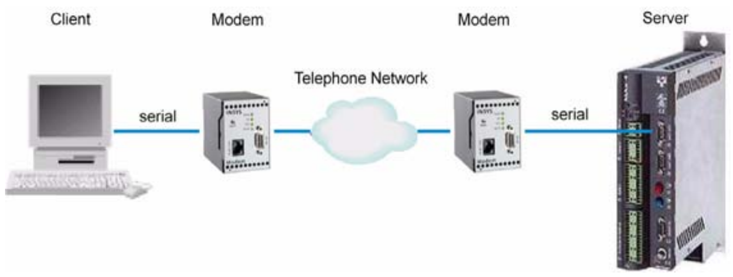
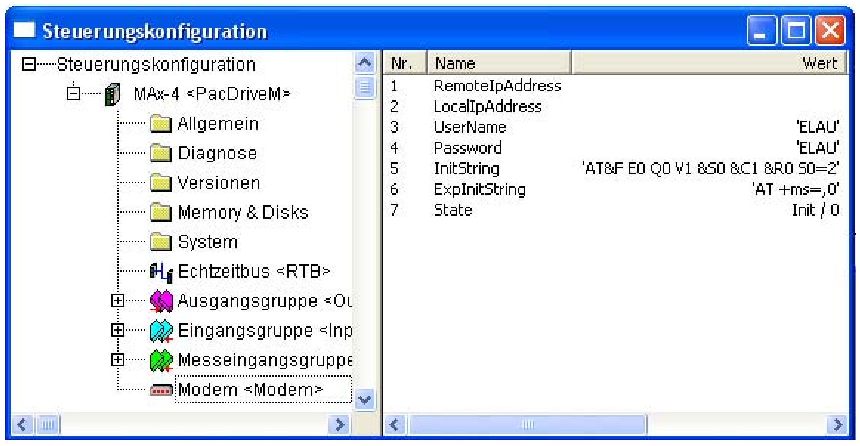


Fig. 7-3: Direct telecommunication connection

You can use an analog or ISDN modem. If you use an ISDN modem, make sure that not the normal ISDN driver, but an analog modem emulator is active on the client computer. The connection to the server requires an analog modem, as ISDN is not yet supported.

Again, the TCP/IP protocol must be installed on both sides. Client and controller must have an IP address, which the user can assign himself. Only those two computers are connected, so that you need not check if those IP addresses are already being used by other computers.

The modem is connected to the serial port COM1 of the MAx-4 PacDrive Controller. To start the PPP server, which controls the connection in the PacDrive Controller, a project containing a modem object must be loaded and the parameter *Com1User* of the MAx-4 PacDrive Controller must have the value “Modem / 1“ or “Auto / 2“.



Nr.	Name	Wert
1	RemoteIpAddress	
2	LocalIpAddress	
3	UserName	'ELAU'
4	Password	'ELAU'
5	InitString	'AT&F E0 Q0 V1 &S0 &C1 &R0 S0=2'
6	ExpInitString	'AT +ms=,0'
7	State	Init / 0

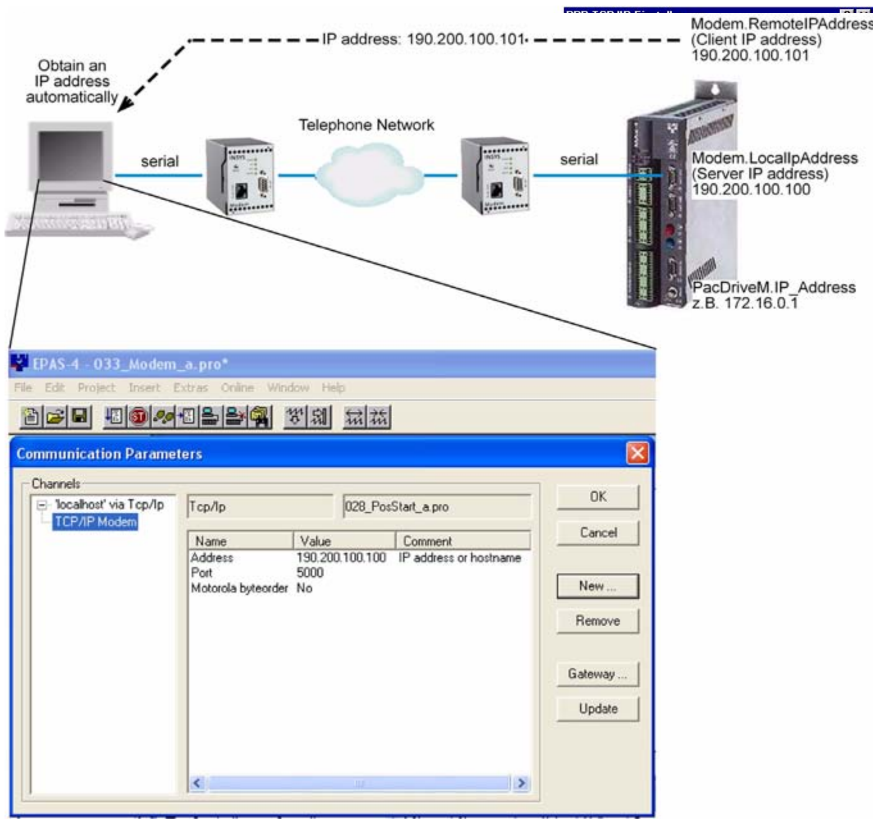
Bild 7-4: Configuration of the modem

The modem is then recognized automatically when the PacDrive controller is booted, as long as it is switched on when booting and connected to the COM port.

If no modem which is on is detected, the COM port is released for serial EPAS-4 communication.
See also „Settings in the Project for the PacDrive Controller“, page 162.

The MAx-4 PacDrive Controller monitors the serial port until the modem answers an incoming call and then establishes a connection.

The call is initialized by the telecommunication network of the remote PC. When dialling up, the user has to enter the user name and the password entered in the control configuration for the Modem object in order to be able to establish a connection. If user name and password are correct, the connection is established. On the remote PC, it can be treated like a normal TCP/IP connection to the MAx-4 PacDrive Controller.


Obtain an IP address automatically
IP address: 190.200.100.101
Modem.RemoteIPAddress (Client IP address) 190.200.100.101
Telephone Network
serial
serial
Modem.LocalIpAddress (Server IP address) 190.200.100.100
PacDriveM.IP_Address z.B. 172.16.0.1
EPAS-4 - 033_Modem_a.pro*
File Edit Project Insert Extras Online Window Help
Communication Parameters
Channels
OK
Cancel
New ...
Remove
Gateway ...
Update

'localhost' via Tcp/Ip

TCP/IP Modem

Tcp/Ip | 028_PosStart_a.pro

Name	Value	Comment
Address	190.200.100.100	IP address or hostname
Port	5000	
Motorola byteorder	No	

Fig. 7-5: Overview of the assignment of TCP/IP addresses

On the client side, a communication network must be installed to access the server. You can set up the network with the help of an assistant. For detailed instructions, please consult the Windows online help. We will only explain the most important settings. You need to enter the telephone number of the connected modem on the server and its IP address.

When the client dials up the server via the modem, a username and password is requested for security reasons. Those parameters can be configured in EPAS-4.

7.2.2 Selection of a Suitable Modem

The modem used for the connection to the MAx-4 PacDrive Controller must be Hayes-compatible and support the AT command set. Nevertheless, you have to check if the following settings are entered in the standard configuration:

- auto-answer function (AT S0=2 - modem answers call after 2nd ring)

- DCD signal processing (AT &C1 - DCD signal follows carrier state)
- hardware handshake on (AT &R0 - modem reacts to CTS signal)
- echo function off (AT E0)
- modem state messages on (AT Q0)
- verbose modem state messages (AT V1)

The serial interface for the modem communication has the following settings:

- baud rate: 38400
- data bits: 8
- stop bits: 1
- parity: none

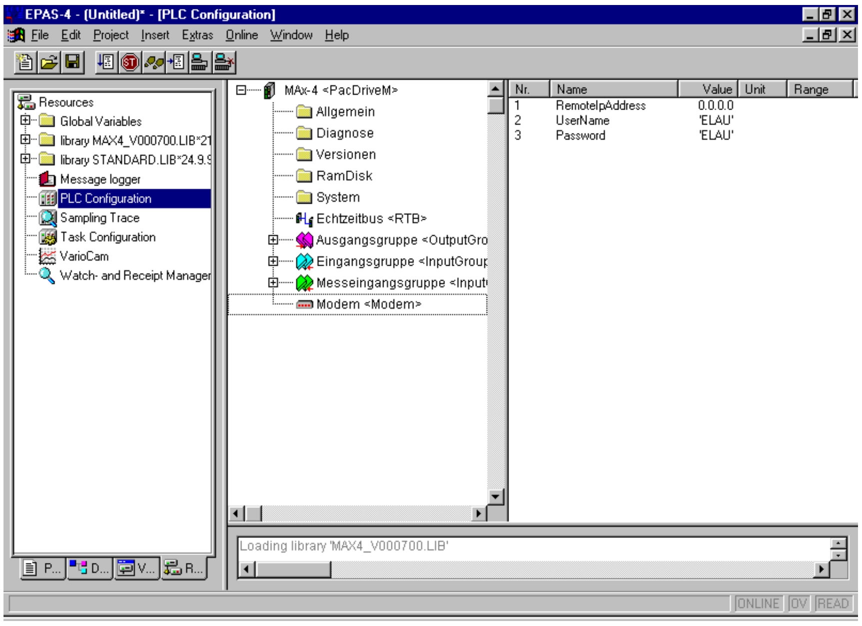
When starting, the MAx-4 PacDrive Controller tries to initialize the modem accordingly. This is done via the *InitString* parameter of the modem. The appropriate AT commands for the basic settings must be entered here. For most modems, the standard init string should already contain the necessary commands. However, you should check in the modem manual if these commands are the same as those of the modem. If necessary, the parameter has to be adjusted. The modem should automatically detect the settings of the serial interface. If automatic detection is not possible, you will have to make the necessary settings manually. For further information, consult the modem manual. Usually settings can be changed by means of DIP switch or with the help of a terminal program such as Hyperterminal (part of *Windows*).

ELAU recommends:

- INSYS modem 56k (INSYS order number: 110.756.WD03)
- INSYS Modem 144 (INSYS order number: 110.6144.WD03)
- Westermo TD-33 v. 90 telemodem (Westermo order number: 31790010)
- Westermo TD-32 telemodem (Westermo order number: 3178-0040)

7.2.3 Settings in the Project for the PacDrive Controller

➢ Enter the object “Modem“ in the control configuration.


EPAS-4 - (Untitled)* - [PLC Configuration]
File Edit Project Insert Extras Online Window Help
Resources
Global Variables
library MAX4_V000700.LIB*21
library STANDARD.LIB*24.9.9
Message logger
PLC Configuration
Sampling Trace
Task Configuration
VarioCam
Watch- and Receipt Manager
P... D... V... R...
MAx-4 <PacDriveM>
Allgemein
Diagnose
Versionen
RamDisk
System
Echtzeitbus <RTB>
Ausgangsgruppe <OutputGro
Eingangsgruppe <InputGroup
Messeingangsgruppe <Input
Modem <Modem>
ONLINE OV READ

Nr.	Name	Value	Unit	Range
1	RemoteIpAddress	0.0.0.0		
2	UserName	'ELAU'		
3	Password	'ELAU'		

Loading library 'MAX4_V000700.LIB'

Fig. 7-6: Object “Modem“ in the control configuration

The seven parameters of the modem object have the following meanding:

RemoteIpAddress

IP address for the computer that connects via modem to the MAx-4 PacDrive Controller. The MAx-4 PacDrive Controller automatically assigns this IP address to the computer when establishing the connection. By default, this address is set to 190.200.100.101.

LocalIpAddress

IP address assigned to the MAx-4 when establishing a modem connection. This IP address is valid for the PPP adapter of the MAx-4 only. By default, this address is set to 190.200.100.100. It cannot be changed.

UserName

User name that must be entered when logging in to establish a connection.

Password

Password that must be entered when logging in to establish a connection.

InitString

The init string contains a sequence of AT commands that are sent to the modem during the initialization phase. In oder to ensure correct modem operation on the MAx-4 PacDrive Controller, some basic settings are required on the modem. The default init string makes these settings and should be valid for most standard modems. However, you should check the AT commands of the modem in the modem manual and, if necessary, change the init string in order to switch on the required functions. The following settings are required:

Default InitString: AT&F E0 Q0 V1 &S0 &C1 &R0 S0=2

Default	Meaning
AT	initiate an AT command
&F	set the modem to the default configuration
E0	switch off the echo function of the modem
Q0	switch on the modem state messages
V1	verbose modem state messages
&S0	DSR signal always 1 if modem ready
&C1	switch on DCD signal (DCD follows the state of the carrier)
&R0	switch on hardware handshake (modem treats the CTS signal)
S0=2	switch on auto-answer function (modem answers after second ring)

Table 7-1: Meaning of the default InitString

ExpInitString

As *InitString* is limited to 40 characters, another 40 charcters can be entered in *ExpInitString* in order to avoid possible modem problems. The latter will be transmitted to the modem in a second round. The string must start with AT, even if no further commands follow. *ExpInitString* is only transmitted to the modem if InitString was processed correctly.

Default-ExpInitString: AT +MS=,0

Value	Meaning
AT	initiate an AT command

Value	Meaning
+MS=,0	switch off automatic baud rate detection This command is only necessary if hand-shake problems occur in the dial-up procedure and may not be available in your modem if it has no Rockwell chip set. The modem recommended by ELAU supports this command.

Table 7-2: Meaning of the default ExpInitString

State

Shows the state of the modem.

Value	Meaning
Init / 0	checking modem
No Modem / 1	no modem connected to COM1 / modem off
Offline / 2	modem in offline mode
Online / 3	modem in online mode (incoming call)
InitString OK / 4	InitString accepted by modem
InitString ERROR / 5	The modem could not be initialized with the init string (maybe the InitString of the modem is faulty or the modem could not be set accordingly). To restart the init procedure, you either have to switch off and on the modem or first change the Com1User parameter of the MAx-4 object via a TCP/IP connection and then write it again.
InitString TIMEOUT / 6	no reply from modem to IniString (see InitString ERROR / 5)
ExpInitString OK / 7	ExpInitString accepted by modem
ExpInitString ERROR / 8	InitString causes error on modem (see InitString ERROR / 5)
ExpInitString TIME-OUT / 9	no reply from modem to ExpInitString (see InitString ERROR / 5)

Table 7-3: Possible values for State

MAx-4.Com1User

Furthermore, there is the parameter *Com1User* of the object “MAx-4“ under “General“ in Meaning. Here, the COM1 port of the MAx-4 PacDrive Controller is set. It has to be set to “Modem / 1“ or “Auto / 2“.
The setting “Modem / 1“ always allocates COM1 to the modem; the COM1 port is checked until a modem is found. The check is done

every 10 seconds.
The setting “Auto / 2“ checks the COM1 port. If a ready modem is detected, that modem is initialized. Otherwise the port is always released for EPAS-4.
In principle, the parameter only has an effect if its value was actually changed in the MAx-4 PacDrive Controller. To re-initialize e.g. a modem, the parameter must first be set from Modem or Auto to None and then reset to the desired value. In the state “Modem / 1“ or “Auto / 2“, the MAx-4 PacDrive Controller also reacts to the removal or connection of devices.

NOTE

The parameter *Com1User* directly influences the serial interface COM1. Therefore this parameter must not be changed by a serial EPAS-4 connection or a modem connection. This would mean that you cut off the connection of the COM port for your own connection, which would result in a serious communication error. In case of remote servicing via modem connection, there must always be a functioning local TCP/IP connection, so that the serial interface can be switched accordingly.

MAx-4.Com1UserState

The parameter Max4.Com1UserState returns the state of the COM1 port.

Value	Meaning
Init / 0	checking for connected devices
EPAS-4 / 1	Com1User is EPAS-4.
Modem -> EPAS-4 / 2	Com1User is Modem; no modem found or modem could not be initialized
Auto -> Search / 3	Com1User is Auto; looking for modem
Modem -> Search / 4	Com1User is Modem; looking for modem
Modem -> Test / 5	Com1User is Modem; initializing detected modem
Modem -> Offline / 6	Com1User is Modem; a modem was found and initialized successfully
Modem -> Online / 7	Com1User is Modem; modem receiving incoming call
None / 8	Com1User is None

Value	Meaning
Auto -> Test / 9	Com1User is Auto; initializing detected modem
Auto -> EPAS-4 / 10	Com1User is Auto; no modem found or modem could not be initialized
Auto -> Modem Offline / 11	Com1User is Auto; a modem was found and initialized successfully
Auto -> Modem Online / 12	Com1User is Auto; modem receiving inco-ming call

Table 7-4: Possible values for MAx-4.Com1UserState

7.2.4 Settings on the Remote PC

To be able to connect a PC to the MAx-4 PacDrive Controller via modem, several prerequisites must be fulfilled:

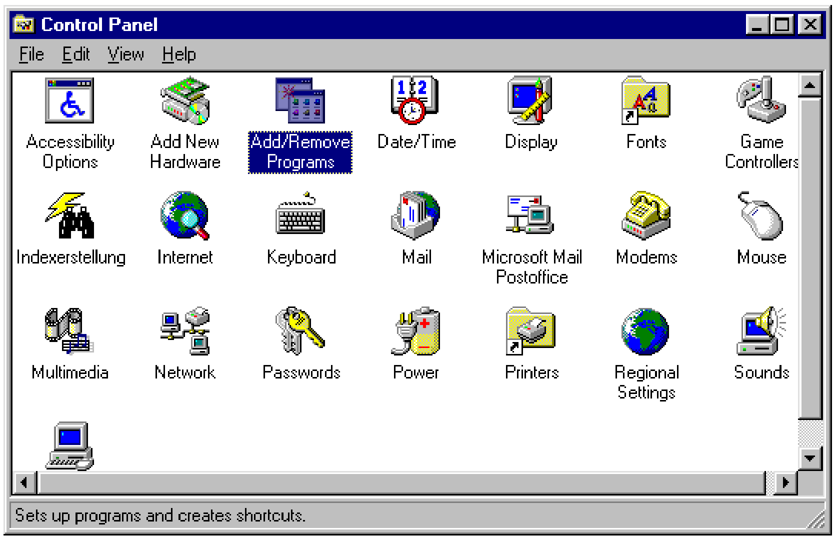
- EPAS-4 installed
- telecommunication network (Windows 95/98) or RAS service (Windows NT) with TCP/IP installed
- modem installed

The installation of EPAS-4 is described in the User Manual EPAS-4. Below, the installation of a telecommunication connection and a modem for Windows 95/98 and Windows NT are described separately.

Windows 95/98

Installation of the telecommunication network

➢ To check if the telecommunication network is already installed, select the menu item Software from the Control Panel.

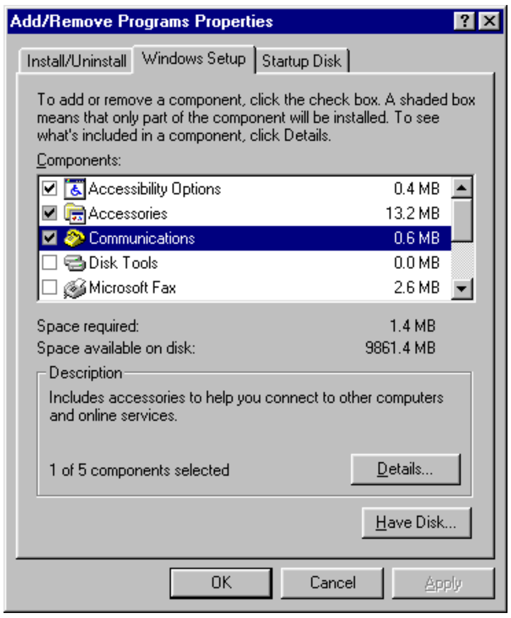


Sets up programs and creates shortcuts.

Fig. 7-7: Control panel Win 95/98

PD_UserMan_Teleservice_us.fm

➢ In the register sheet “Windows Setup“, select the component “Connections“ and click on the button “Details...“ .



Add/Remove Programs Properties

Install/Uninstall | Windows Setup | Startup Disk

To add or remove a component, click the check box. A shaded box means that only part of the component will be installed. To see what's included in a component, click Details.

Components:

☑	Accessibility Options	0.4 MB
☑	Accessories	13.2 MB
☑	Communications	0.6 MB
☐	Disk Tools	0.0 MB
☐	Microsoft Fax	2.6 MB

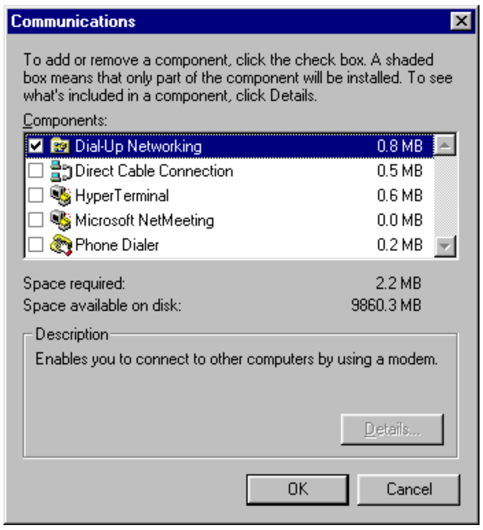
Space required: 1.4 MB
Space available on disk: 9861.4 MB

Description
Includes accessories to help you connect to other computers and online services.

1 of 5 components selected

Fig. 7-8: Properties of Software / Windows Setup

➢ Mark the item Telecommunication network, if it is not marked already.



Communications

To add or remove a component, click the check box. A shaded box means that only part of the component will be installed. To see what's included in a component, click Details.

Components:

☑	Dial-Up Networking	0.8 MB
☐	Direct Cable Connection	0.5 MB
☐	HyperTerminal	0.6 MB
☐	Microsoft NetMeeting	0.0 MB
☐	Phone Dialer	0.2 MB

Space required: 2.2 MB
Space available on disk: 9860.3 MB

Description
Enables you to connect to other computers by using a modem.

Fig. 7-9: Connections

➢ Confirm your selection by clicking on the button “OK“.

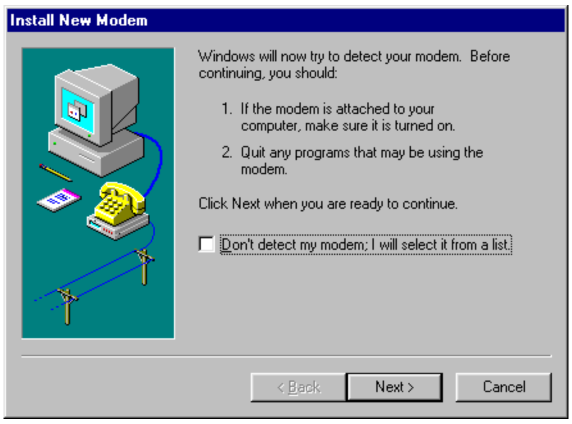
This starts the installation of the telecommunication network, which requires your Windows installation CD. The telecommunication network will be available after restarting your PC. As the TCP/IP protocol will be installed in the next step, you do not need to restart your PC at this stage, because another restart will be necessary after the TCP/IP installation.

Installation of the TCP/IP protocol

- If the TCP/IP protocol is not yet installed on your PC, you will have to do it now. The installation is described in the Operating Manual EPAS-4.

Configuration of telecommunication connection and modem

- Start the telecommunication connection (Start -> Programs -> Accessories -> Telecommunication Network for Win95; Start -> Programs -> Accessories -> Comminication -> Telecommunication Network for Win98). When you start for the first time or select Create new connection, an assistant will appear and take you through the configuration of your connection. If no modem has been installed, this will be done now.


Install New Modem

Windows will now try to detect your modem. Before continuing, you should:

1. If the modem is attached to your computer, make sure it is turned on.
2. Quit any programs that may be using the modem.

Click Next when you are ready to continue.

☐ Don't detect my modem; I will select it from a list.

Fig. 7-10: Dialog: Install new modem

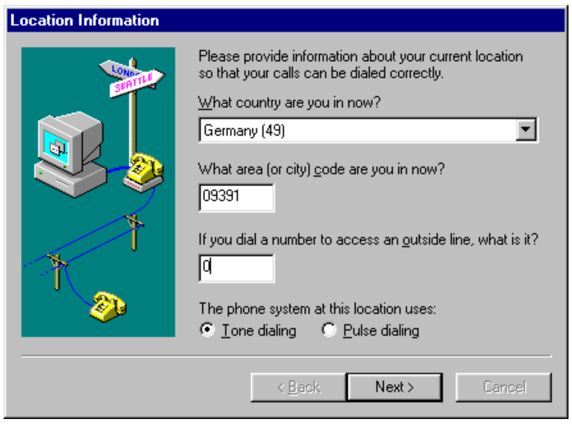
You have the choice to have the modem detected automatically or to select it from a list.

- Clicking on “Continue“ will take you through the modem installation procedure.

NOTE

In case of installation problems, please consult the manual of your modem.

- Enter the location settings for the modem.


Location Information

Please provide information about your current location so that your calls can be dialed correctly.

What country are you in now?
Germany (49)

What area (or city) code are you in now?
09391

If you dial a number to access an outside line, what is it?
0

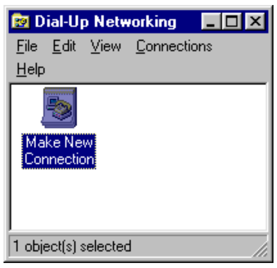
The phone system at this location uses:
(●) Tone dialing ( ) Pulse dialing

Fig. 7-11: Location settings

The modem is now installed and you can establish a connection from the PC to the MAx-4 PacDrive Controller using "Create new connection".

Establishing a connection between the PC and the MAx-4 PacDrive Controller

- In the telecommunication network, select "Create new connection".

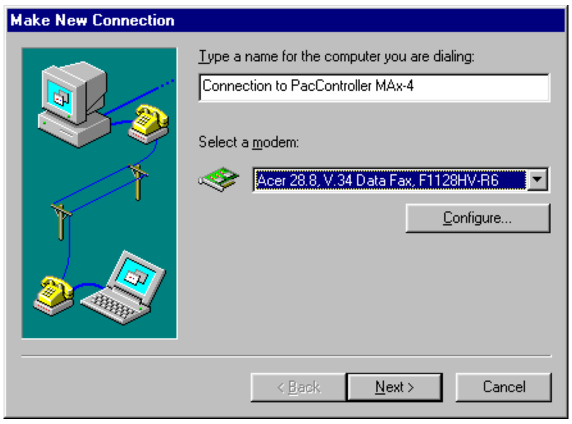


1 object(s) selected

Fig. 7-12: Telecommunication network / Create new connection

- Give the connection a name and select the desired modem. Then click on the button "Configure..." to make the modem settings.

PD_UserMan_Teleservice_us.fm

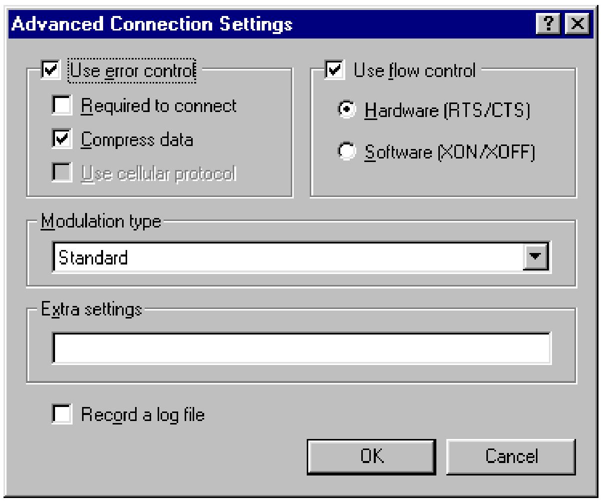

Make New Connection

Type a name for the computer you are dialing:
Connection to PacController MAx-4

Select a modem:
Acer 28.8, V.34 Data Fax, F1128HV-R6

Fig. 7-13: Create new connection

- Select the register sheet “Settings“. Please check the connection settings for the following values:
  – data bits: 8
  – parity: none
  – stop bits:1
- Under “Extended modem settings“, *error control* and *data compression* should be marked and “Hardware handshake (RTS/CTS)“ should be set as *Data flow control*.


Advanced Connection Settings

[✓] Use error control
[ ] Required to connect
[✓] Compress data
[ ] Use cellular protocol

[✓] Use flow control
[● Hardware (RTS/CTS)]
○ Software (XON/XOFF)

Modulation type: Standard

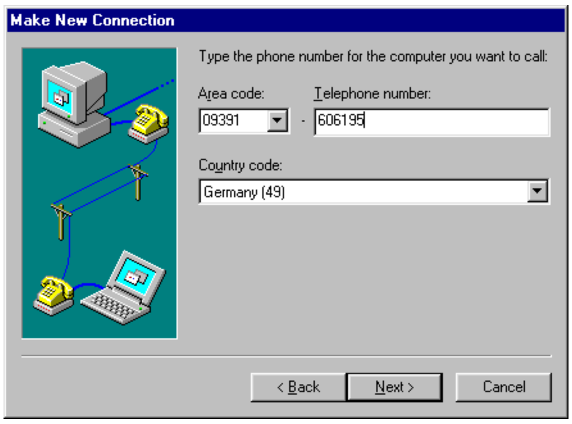
Extra settings:

[ ] Record a log file

Fig. 7-14: Extended modem settings

- After clicking on OK twice, you can terminate the first step of the assistant with “Continue“.

- In the second step, enter the telephone number at which the MAx-4 PacDrive Controller can be reached and confirm with “Continue“.



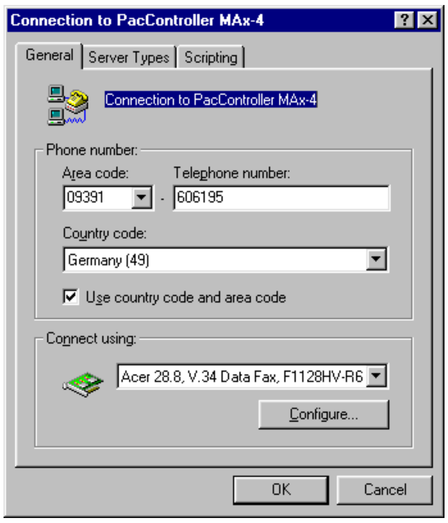
Type the phone number for the computer you want to call:

Area code: 09391 - Telephone number: 606195

Country code: Germany (49)

Fig. 7-15: Create a new connection

- Then the connection is created by clicking on the button Finish.
- Call up the properties dialog of the connection by clicking with the right mouse button and selecting “Properties“ from the context menu.



Connection to PacController MAx-4

Phone number:
Area code: 09391 - Telephone number: 606195

Country code: Germany (49)

☑ Use country code and area code

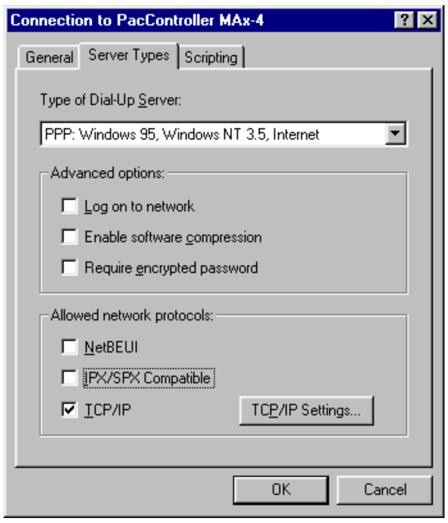
Connect using: Acer 28.8, V.34 Data Fax, F1128HV-R6

Fig. 7-16: Connection with MAx-4 PacDrive Controller / General

- Select the register “Server Types“ and make the following settings:
  – Type of telecommunication server: PPP: Internet, Windows

NT Server, Windows 95 or Windows 98

– Deactivate all other options
– Activate only TCP/IP as admissible network protocols



Connection to PacController MAx-4

General | Server Types | Scripting

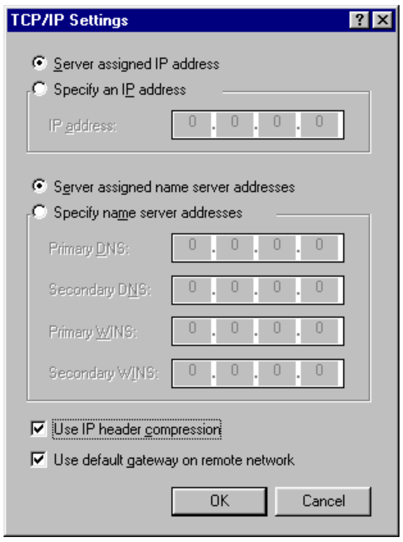
Type of Dial-Up Server: PPP: Windows 95, Windows NT 3.5, Internet

Advanced options:
- [ ] Log on to network
- [ ] Enable software compression
- [ ] Require encrypted password

Allowed network protocols:
- [ ] NetBEUI
- [ ] IPX/SPX Compatible
- [✓] TCP/IP

Fig. 7-17: Connection with MAx-4 PacDrive Controller / Server types

➢ Now make the TCP/IP settings by clicking on the corresponding buttons.
  – mark the IP address assigned by the server
  – activate the name server adress assigned by the server
  – activate IP header compression
  – activate standard gateway in the remote network



TCP/IP Settings

[✓ Server assigned IP address]
Specify an IP address

IP address: 0 . 0 . 0 . 0

[✓ Server assigned name server addresses]
Specify name server addresses

Primary DNS: 0 . 0 . 0 . 0
Secondary DNS: 0 . 0 . 0 . 0
Primary WINS: 0 . 0 . 0 . 0
Secondary WINS: 0 . 0 . 0 . 0

- [✓] Use IP header compression
- [✓] Use default gateway on remote network

Fig. 7-18: PPP TCP / IP settings

- Click on OK
- The connection is now available.

Use standard gateway for remote network

Indicates if the standard gateway is used in the remote network for the duration of the connection. When you create a telecommunication or VPN connection, a new standard route using the connection to the RAS server is added to the IP routing table. This standard route receives the lowest metrics. If a standard route already exists, it remains in the routing table, but receives higher metrics. If a new standard route with lower metrics is added, locations that could previously be reached via the original standard route may no longer be reachable.

A computer with this operating system in a firmware network has e.g. a LAN connection in that network and is configured with a standard gateway of the local IP router. If the option **Use standard gateway for remote network** is selected, the standard gateway and not the local company router becomes the Internet service provider when the user calls up an Internet service provider. This is why all locations in the corporate network, except those in the local network, cannot be reached for the duration of the connection to the Internet service provider. If the option **Use standard gateway for remote network** is not selected, the original standard route remains unchanged, no new standard route is added and the locations on the Internet are not available.

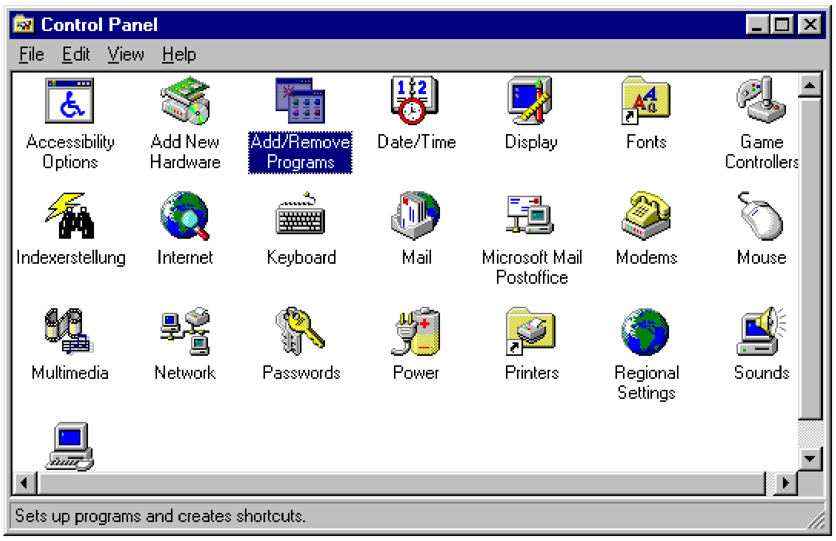
Windows NT 4.0

<u>Set up TCP / IP protocol</u>

- If the TCP/IP protocol has not yet been installed on your PC, you have to do it now. The installation is described in the Operating Manual EPAS-4.

<u>Set up the RAS service (Remote Access Service)</u>

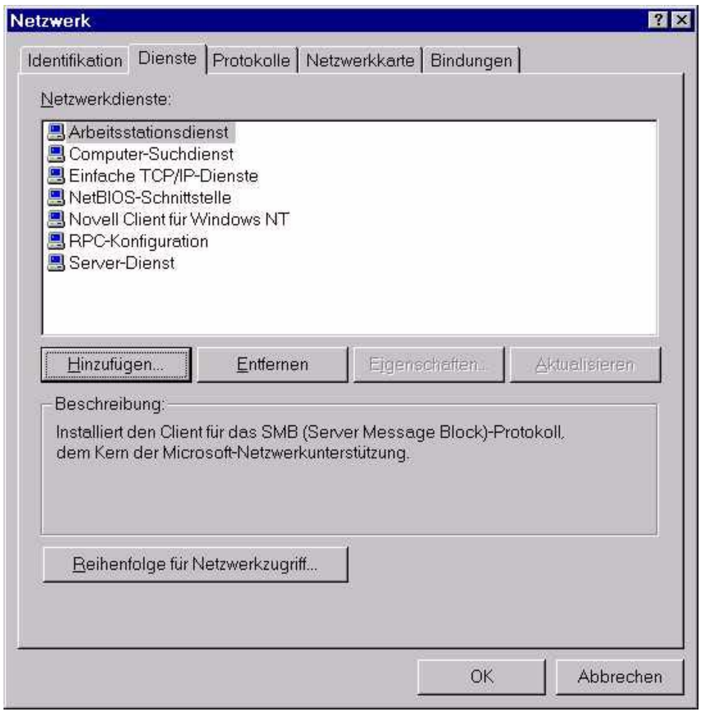
➢ From the Control panel, select “Network“.



Sets up programs and creates shortcuts.

Fig. 7-19: Control panel Win NT

➢ Check in the dialog box “Network“ in the register “Services“ if the RAS service exists. If not, you have to install it using the button “Add...“.

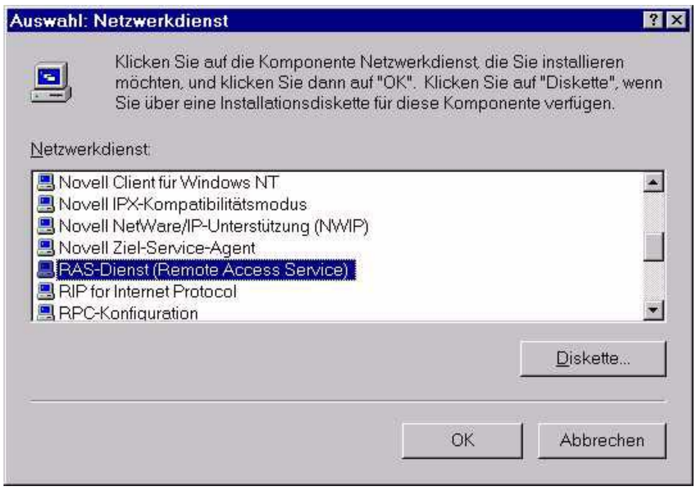


Arbeitsstationsdienst
Computer-Suchdienst
Einfache TCP/IP-Dienste
NetBIOS-Schnittstelle
Novell Client für Windows NT
RPC-Konfiguration
Server-Dienst

Installiert den Client für das SMB (Server Message Block)-Protokoll, dem Kern der Microsoft-Netzwerkunterstützung.

Fig. 7-20: Control panel Win NT / Network / Services

➢ Select RAS service and click on “OK“.



Auswahl: Netzwerkdienst

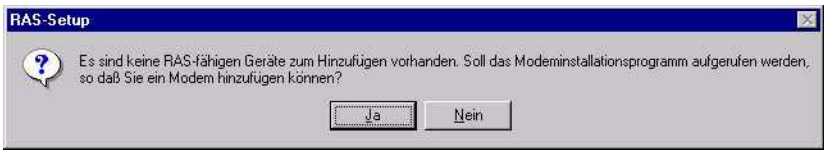
Klicken Sie auf die Komponente Netzwerkdienst, die Sie installieren möchten, und klicken Sie dann auf "OK". Klicken Sie auf "Diskette", wenn Sie über eine Installationsdiskette für diese Komponente verfügen.

Netzwerkdienst:

- Novell Client für Windows NT
- Novell IPX-Kompatibilitätsmodus
- Novell NetWare/IP-Unterstützung (NWIP)
- Novell Ziel-Service-Agent
- RAS-Dienst (Remote Access Service)
- RIP for Internet Protocol
- RPC-Konfiguration

Fig. 7-21: Control panel Win NT / Network / RAS serivce

➢ If no modem has been installed, the RAS setup will ask if you want to call up the modem installation program. Confirm with “Yes“.



RAS-Setup

Es sind keine RAS-fähigen Geräte zum Hinzufügen vorhanden. Soll das Modeminstallationsprogramm aufgerufen werden, so daß Sie ein Modem hinzufügen können?

Fig. 7-22: RAS setup

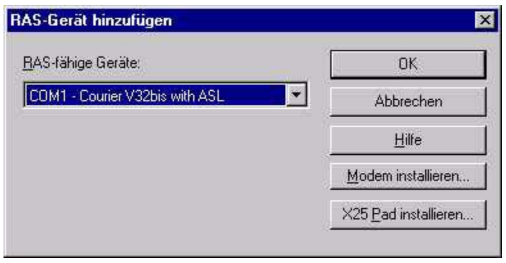
You have the choice to have the moden detected automatically or selet it from a list.

➢ Clicking on the button “Continue“ will take you through the installation procedure.

NOTE

In case of installation problems, please consult the manual of your modem.

➢ Now the RAS service should detect the modem installed and and select it (confirm with OK).

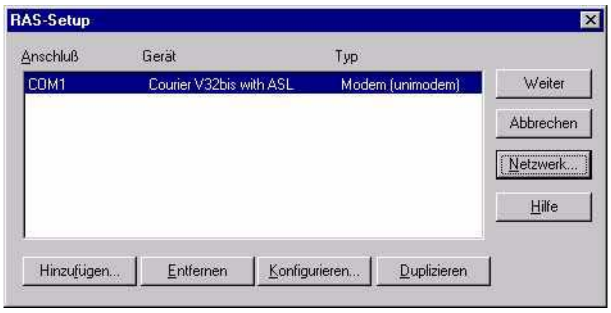


RAS-Gerät hinzufügen

RAS-fähige Geräte: COM1 - Courier V32bis with ASL

Fig. 7-23: Add RAS device

➢ Now select a connection in the RAS setup and click on the button "Network...".

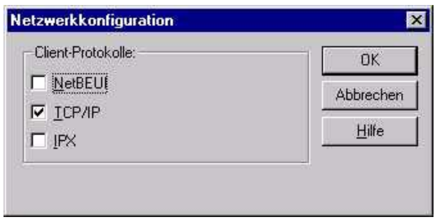


RAS-Setup

Anschluß	Gerät	Typ
COM1	Courier V32bis with ASL	Modem (unimodem)

Fig. 7-24: RAS setup

➢ Mark only TCP/IP as client protocols and confirm your selection with "OK".



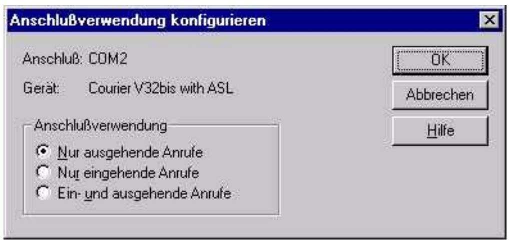
Netzwerkkonfiguration

Client-Protokolle:

- ☐ NetBEUI
- ☑ TCP/IP
- ☐ IPX

Fig. 7-25: Newtork configuration

- Now you need to check under “Configure...“ if the setting is „outgoing calls only“. (If the PC is to take calls for other purposes, you can also mark incoming and outgoing calls).


Anschlußverwendung konfigurieren

Anschluß: COM2

Gerät: Courier V32bis with ASL

Anschlußverwendung

(●) Nur ausgehende Anrufe
( ) Nur eingehende Anrufe
( ) Ein- und ausgehende Anrufe

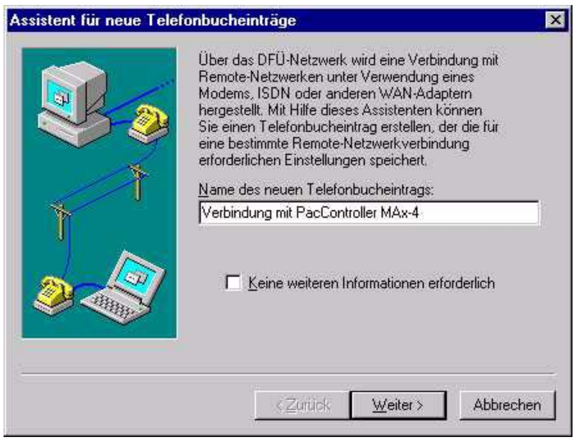
Fig. 7-26: Configure connection purpose

- Confirm your selection with “OK“
- Select “Continue“ to complete the configuration.

Now the RAS service is available.

<u>Establishing a connection between PC and MAx-4 PacController</u>

- Start the telecommunication network under Start, Programs, Accessories, Telecommunication network. You will be asked to create a new directory entry (confirm with OK).


Assistent für neue Telefonbucheinträge

Über das DFÜ-Netzwerk wird eine Verbindung mit Remote-Netzwerken unter Verwendung eines Modems, ISDN oder anderen WAN-Adaptern hergestellt. Mit Hilfe dieses Assistenten können Sie einen Telefonbucheintrag erstellen, der die für eine bestimmte Remote-Netzwerkverbindung erforderlichen Einstellungen speichert.

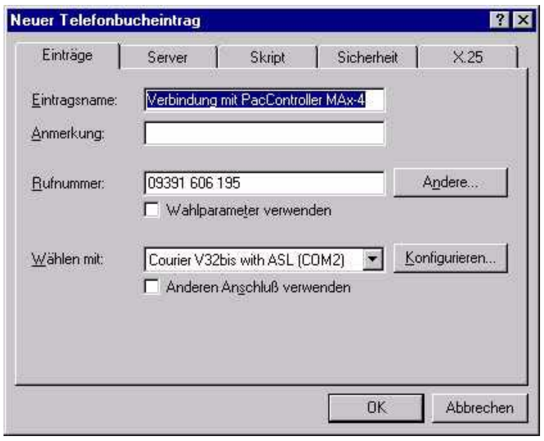
Name des neuen Telefonbucheintrags: Verbindung mit PacController MAx-4

☐ Keine weiteren Informationen erforderlich

Fig. 7-27: Assistant for new directory entries

- Give the connection a name, mark the checkbox “No further information required“ and confirm by clicking on “Finish“. This will take you to the configuration dialog box for the new directory entry.

- In the register “Entries“ you can enter the telephone number at which the MAx-4 PacDrive Controller can be reached.



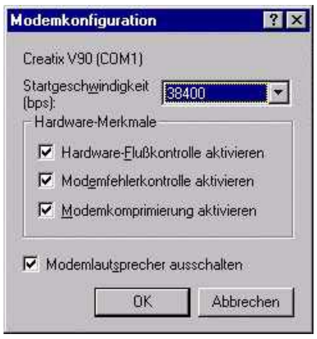
Neuer Telefonbucheintrag

Einträge | Server | Skript | Sicherheit | X.25

Eintragsname: Verbindung mit PacController MAx-4
Anmerkung:
Rufnummer: 09391 606 195
☐ Wahlparameter verwenden
Wählen mit: Courier V32bis with ASL (COM2)
☐ Anderen Anschluß verwenden

Fig. 7-28: New directory entry

- Click on the button “Configure...“ to open the configuration dialog for hte modem and activate all hardware properties.



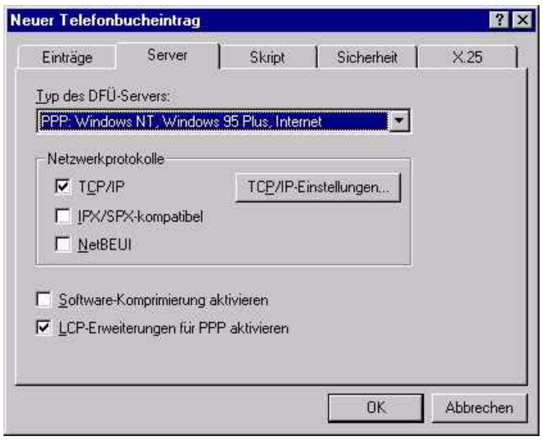
Modemkonfiguration

Creatix V90 (COM1)
Startgeschwindigkeit (bps): 38400
Hardware-Merkmale
☑ Hardware-Flußkontrolle aktivieren
☑ Modemfehlerkontrolle aktivieren
☑ Modemkomprimierung aktivieren
☑ Modemlautsprecher ausschalten

Fig. 7-29: Modem configuration

- Confirm your selection with “OK“.
- In the register sheet Server, make the following settings:
  – type of telecommunication server: PPP: Windows NT, Win-

dows 95 or 98, Internet
- activate only TCP/IP as network protocols
- deactivate Activate Software compression
- activate Activate LCP extensions for PPP



Neuer Telefonbucheintrag

Einträge | Server | Skript | Sicherheit | X.25

Typ des DFÜ-Servers: PPP: Windows NT, Windows 95 Plus, Internet

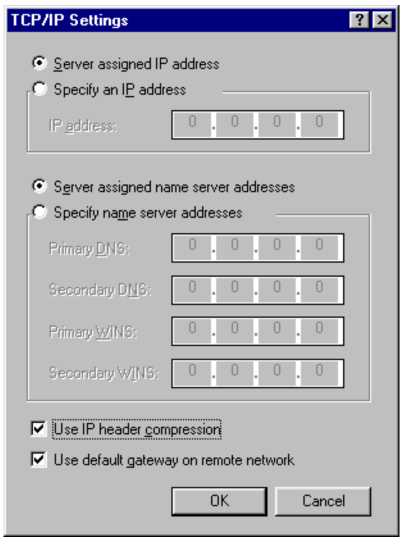
Netzwerkprotokolle
- [✓] TCP/IP (TCP/IP-Einstellungen...)
- [ ] IPX/SPX-kompatibel
- [ ] NetBEUI

- [ ] Software-Komprimierung aktivieren
- [✓] LCP-Erweiterungen für PPP aktivieren

OK | Abbrechen

Fig. 7-30: New directory entry / Server

➢ The TCP/IP settings are made by clicking on the corresponding buttons.
- mark the setting IP address allocated by server
- activate Name server addresses allocated by server
- activate IP header compression
- activate Use standard gateway in remote network



TCP/IP Settings

- [✓ Server assigned IP address]
- Specify an IP address

IP address: 0 . 0 . 0 . 0

- [✓ Server assigned name server addresses]
- Specify name server addresses

Primary DNS: 0 . 0 . 0 . 0
Secondary DNS: 0 . 0 . 0 . 0
Primary WINS: 0 . 0 . 0 . 0
Secondary WINS: 0 . 0 . 0 . 0

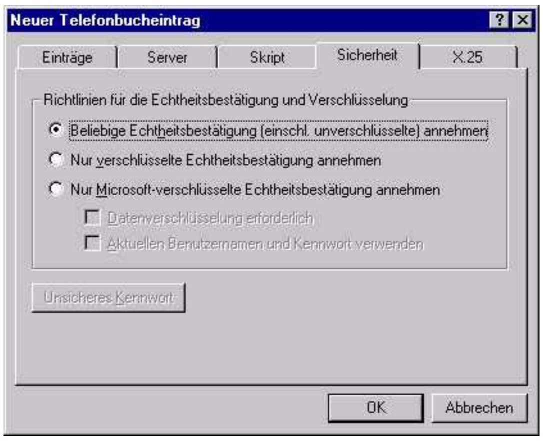
- [✓] Use IP header compression
- [✓] Use default gateway on remote network

OK | Cancel

Fig. 7-31: TCP/IP settings

➢ Confirm with “OK“.

- In the register sheet “Safety“, you have to mark the setting “Accept any authentification (even non-encoded).



Neuer Telefonbucheintrag

Einträge | Server | Skript | Sicherheit | X.25

Richtlinien für die Echtheitsbestätigung und Verschlüsselung

- [✓ Beliebige Echtheitsbestätigung (einschl. unverschlüsselte) annehmen]
- Nur verschlüsselte Echtheitsbestätigung annehmen
- Nur Microsoft-verschlüsselte Echtheitsbestätigung annehmen
  - Datenverschlüsselung erforderlich
  - Aktuellen Benutzernamen und Kennwort verwenden

Fig. 7-32: New directory entry / Safety

- Confirm your selection with “OK“

The connection is now available for selection.

Windows 2000

<u>Installation of the TCP/IP protocol</u>

- If the TCP/IP protocol is not installed on your PC, you have to install it now. The installation is described in the EPAS-4 operating manual.

<u>Installation of the modem on the PC</u>

- Open the control panel (Start / Settings / Control panel).
- Select Telephone and modem options.

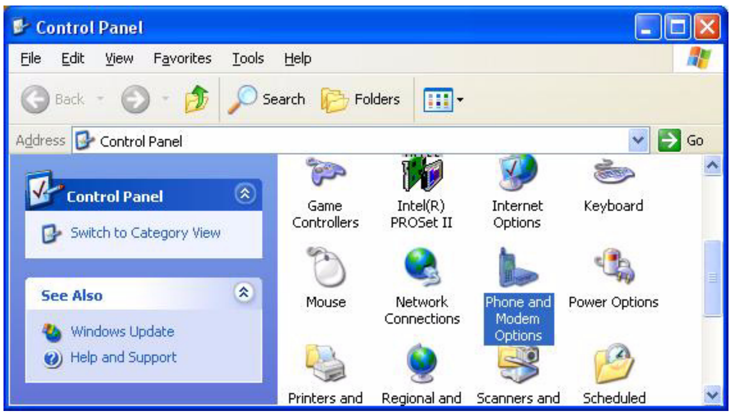
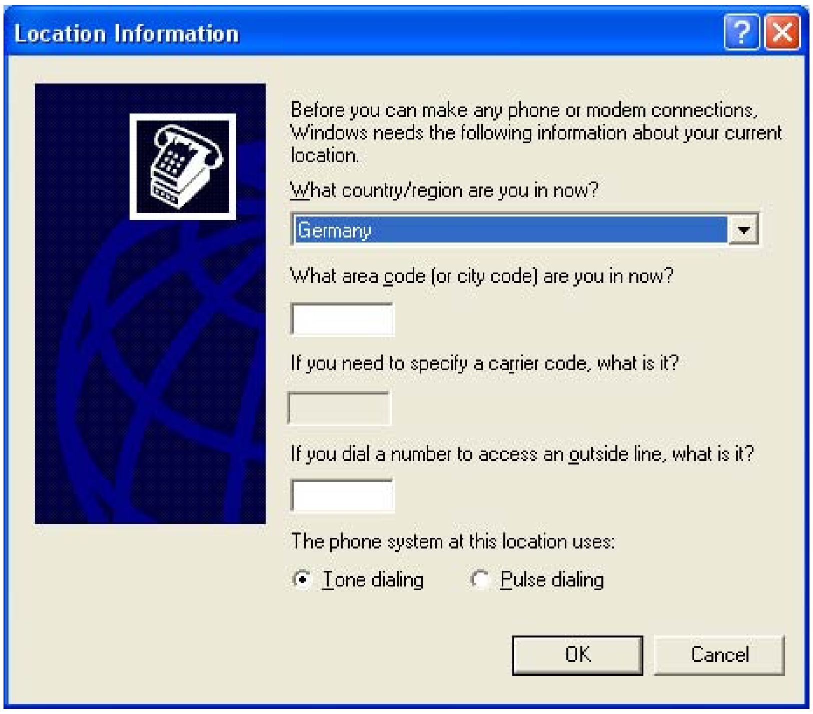


Bild 7-33: Control panel Windows 2000 / XP

- Select Telephone and modem options.


Location Information

Before you can make any phone or modem connections, Windows needs the following information about your current location.

What country/region are you in now? Germany

What area code (or city code) are you in now?

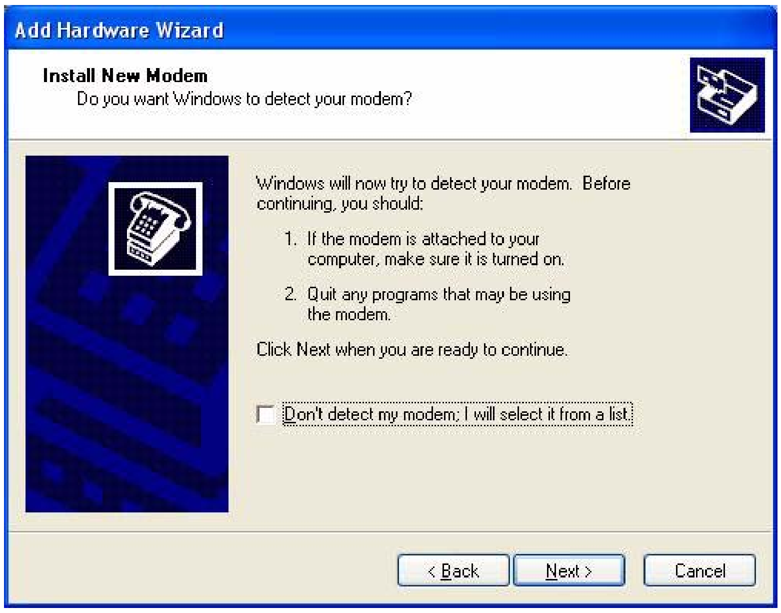
If you need to specify a carrier code, what is it?

If you dial a number to access an outside line, what is it?

The phone system at this location uses: [✓ Tone dialing] Pulse dialing

Bild 7-34: Dialog location information

- Choose in the register Modem „add to...“


Add Hardware Wizard

Install New Modem
Do you want Windows to detect your modem?

Windows will now try to detect your modem. Before continuing, you should:

1. If the modem is attached to your computer, make sure it is turned on.
2. Quit any programs that may be using the modem.

Click Next when you are ready to continue.

☐ Don't detect my modem; I will select it from a list.

Bild 7-35: Hardware Assistant / Win 2000 / XP

- Follow the instructions of the hardware assistant

PD_UserMan_Teleservice_us.fm

NOTE

In case of installation problems, please consult the operating manual of your modem.

<u>Creating a connection with the MAx-4 PacDrive controller</u>

- Start the telecommunication network (Start -> Programs -> Accessories -> Communication -> Network and telecommunication connections).
- In the telecommunication network, select “Create new connection“.

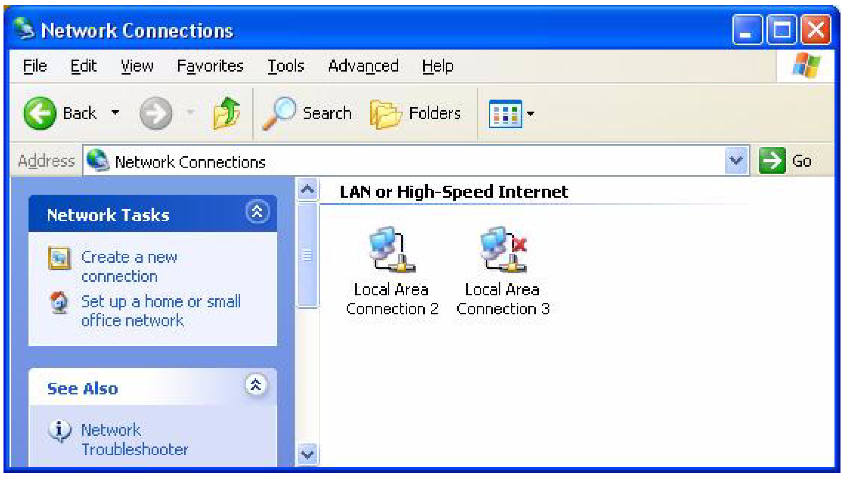
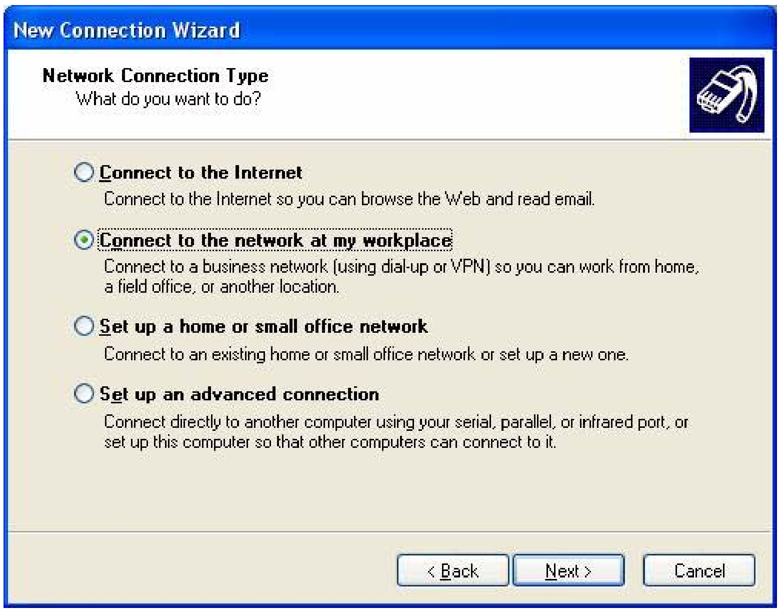


Bild 7-36: Network telecommunication / create new connection


New Connection Wizard

Network Connection Type
What do you want to do?

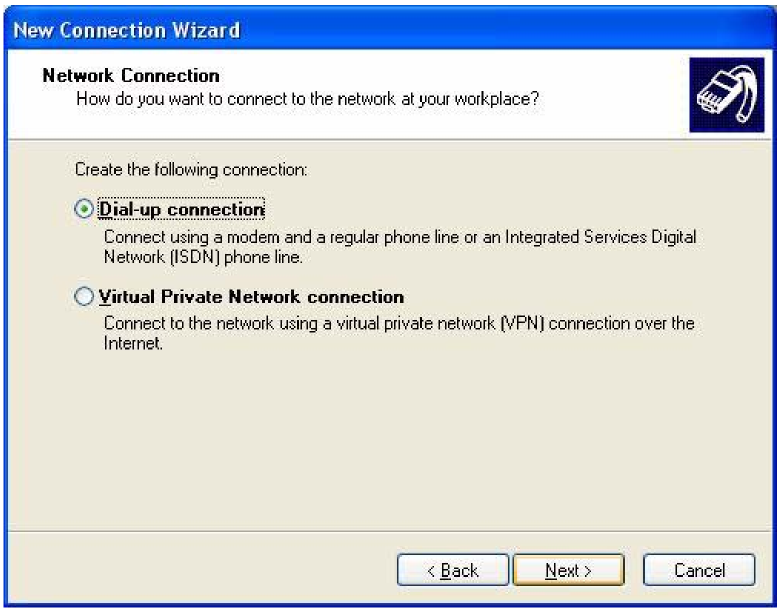
( ) **Connect to the Internet**
Connect to the Internet so you can browse the Web and read email.

(●) **Connect to the network at my workplace**
Connect to a business network (using dial-up or VPN) so you can work from home, a field office, or another location.

( ) **Set up a home or small office network**
Connect to an existing home or small office network or set up a new one.

( ) **Set up an advanced connection**
Connect directly to another computer using your serial, parallel, or infrared port, or set up this computer so that other computers can connect to it.

Bild 7-37: Network connection assistant / 1



Network Connection
How do you want to connect to the network at your workplace?

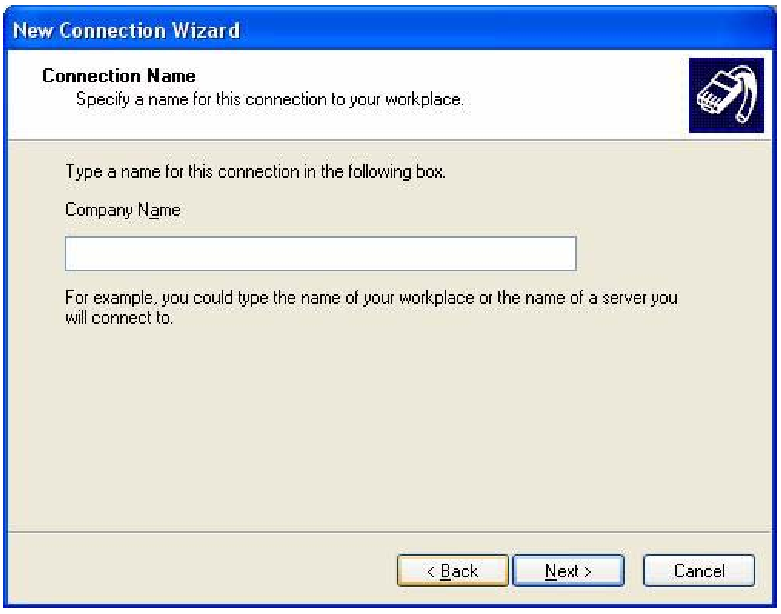
Create the following connection:

[✓ Dial-up connection]
Connect using a modem and a regular phone line or an Integrated Services Digital Network (ISDN) phone line.

Virtual Private Network connection
Connect to the network using a virtual private network (VPN) connection over the Internet.

Bild 7-38: Network connection assistant / 2

.



Connection Name
Specify a name for this connection to your workplace.

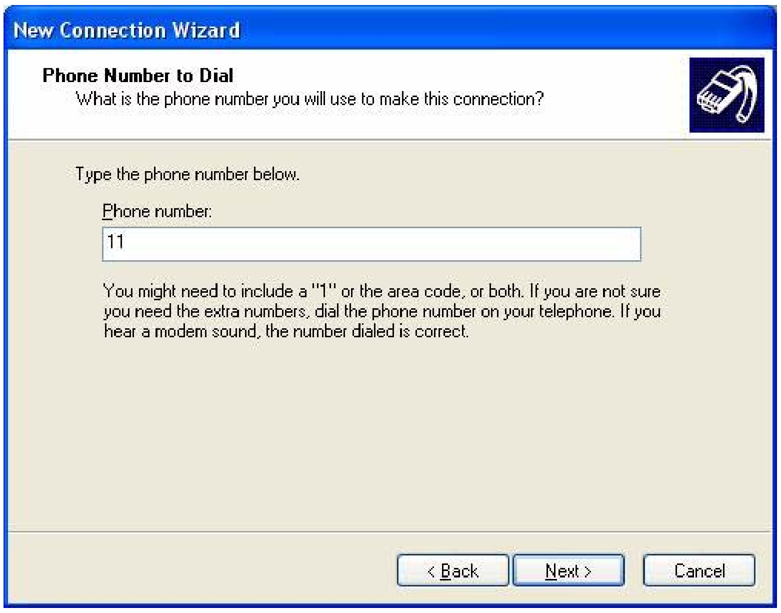
Type a name for this connection in the following box.

Company Name

For example, you could type the name of your workplace or the name of a server you will connect to.

Bild 7-39: Network connection assistant / 3

PD_UserMan_Teleservice_us.fm


New Connection Wizard

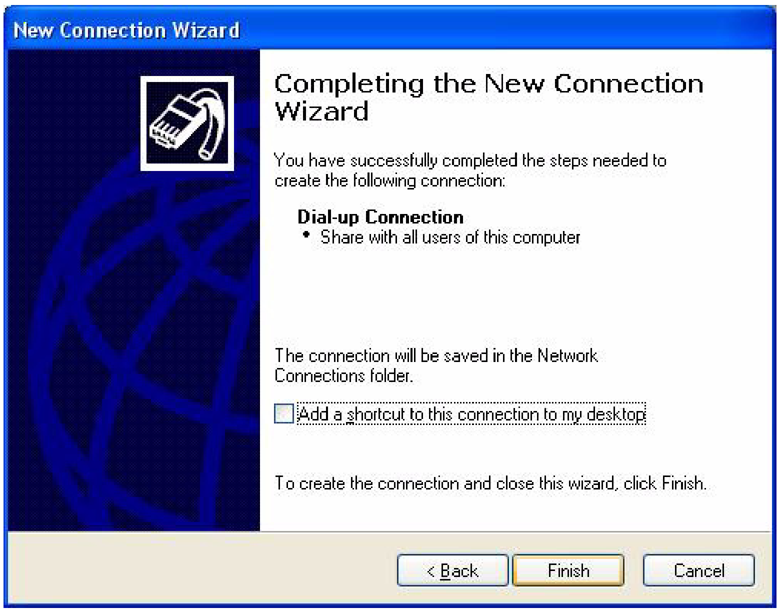
Phone Number to Dial
What is the phone number you will use to make this connection?

Type the phone number below.

Phone number:
11

You might need to include a "1" or the area code, or both. If you are not sure you need the extra numbers, dial the phone number on your telephone. If you hear a modem sound, the number dialed is correct.

Bild 7-40: Network connection assistant / 4


New Connection Wizard

Completing the New Connection Wizard

You have successfully completed the steps needed to create the following connection:

Dial-up Connection
- Share with all users of this computer

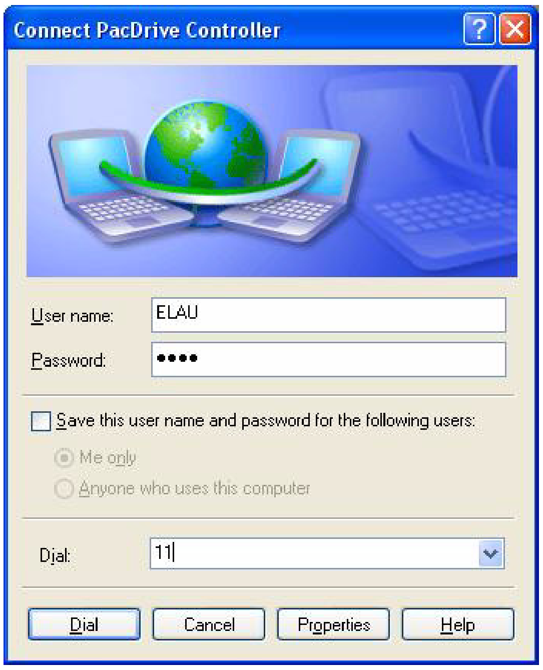
The connection will be saved in the Network Connections folder.

☐ Add a shortcut to this connection to my desktop

To create the connection and close this wizard, click Finish.

Bild 7-41: Network connection assistant / 5

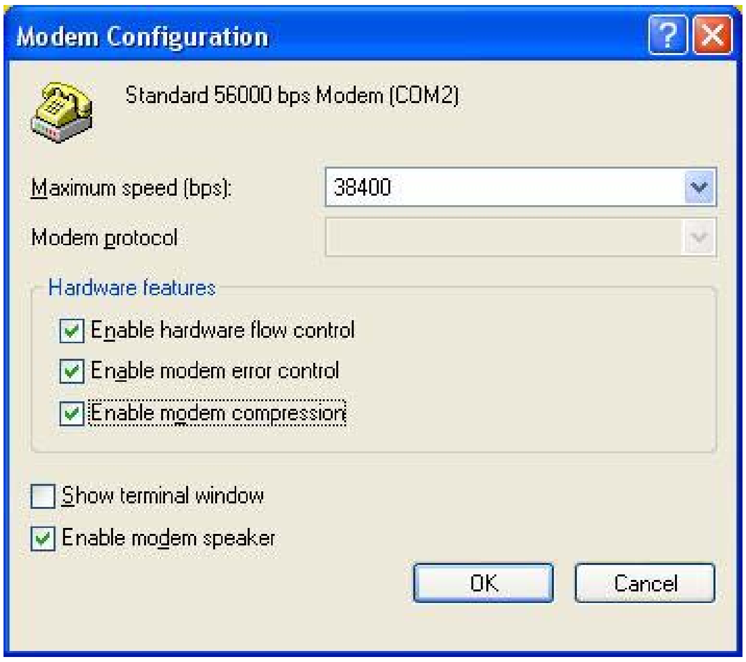
Once you created the new connection, it is available in the folder *Network and telecommunication connections* and can be started from there.



User name: ELAU
Password: ●●●●
☐ Save this user name and password for the following users:
◉ Me only
○ Anyone who uses this computer
Dial: 11

Bild 7-42: Dialog to the connection setup

- To check the settings, select “Properties“.
- In the “General“ tab, select the button “Configure...“ Modem configuration


Modem Configuration

Standard 56000 bps Modem (COM2)

Maximum speed (bps): 38400

Modem protocol

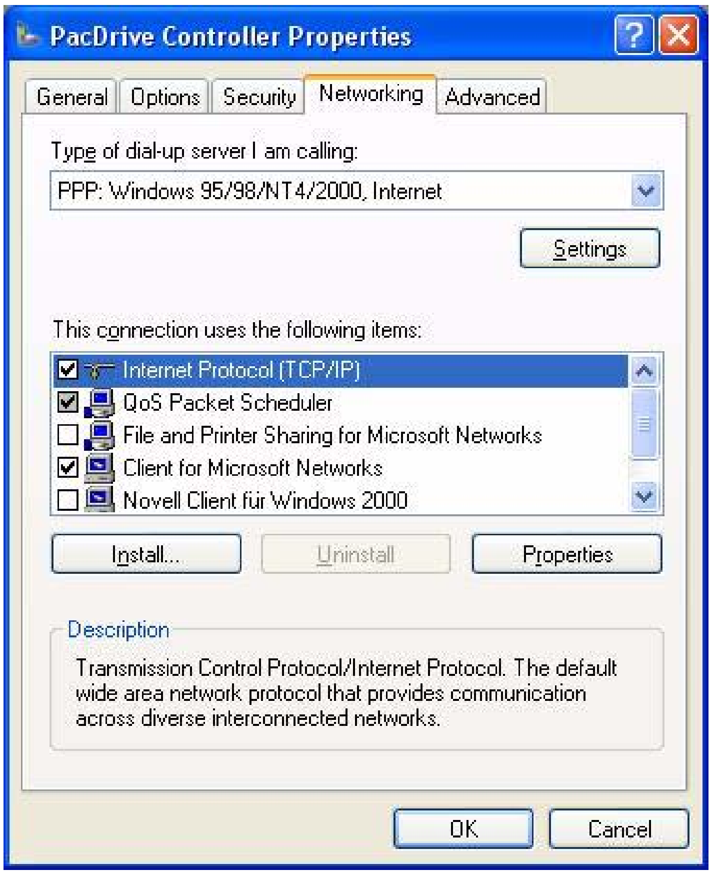
Hardware features

- [✓] Enable hardware flow control
- [✓] Enable modem error control
- [✓] Enable modem compression

- [ ] Show terminal window
- [✓] Enable modem speaker

Bild 7-43: Modem configuration

- Select the hardware options *Activate hardware flow control, Activate modem error controller* and *Activate modem compression*.
- Confirm your selection with “OK“.
- Select the tab “Network“.



Type of dial-up server I am calling:

PPP: Windows 95/98/NT4/2000, Internet

This connection uses the following items:

- [✓] Internet Protocol (TCP/IP)
- [✓] QoS Packet Scheduler
- [ ] File and Printer Sharing for Microsoft Networks
- [✓] Client for Microsoft Networks
- [ ] Novell Client für Windows 2000

Description

Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks.

Bild 7-44: Properties of the connection / network

4 Check the “Type of dial-up server to be called“. *PPP:Windows 95/98/NT4/2000, Internet* must be selected.

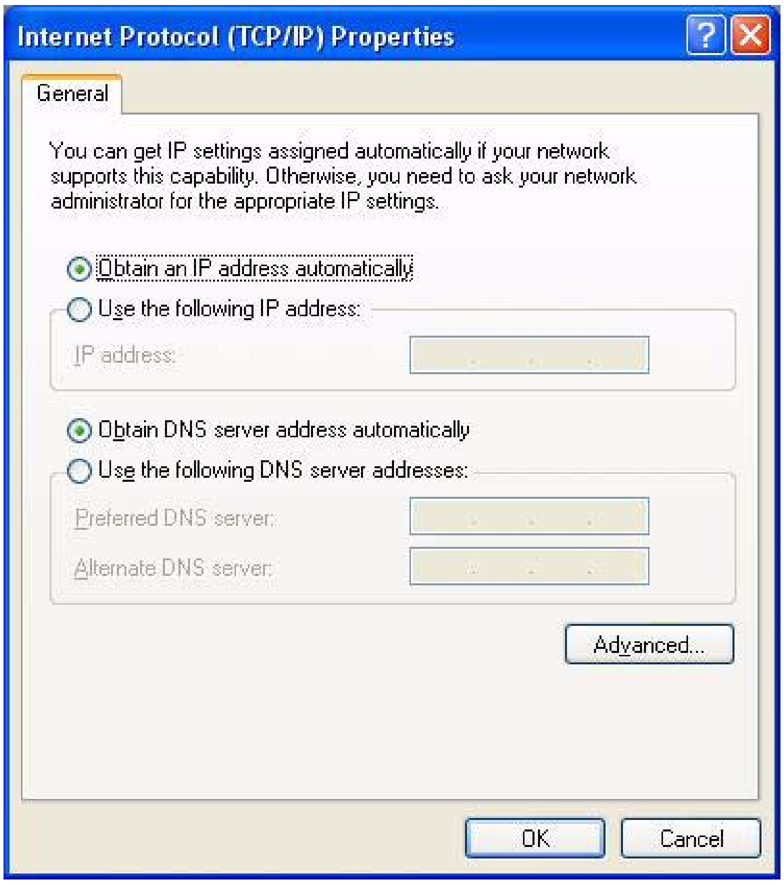
4 Check that *Internet protocol (TCP/IP)* is marked as the only protocol.

4 Select *Internet protocol (TCP/IP)*.

4 Click on the button “Properties“ and select

– *Automatically get IP address*

– *Automatically get DNS server address*



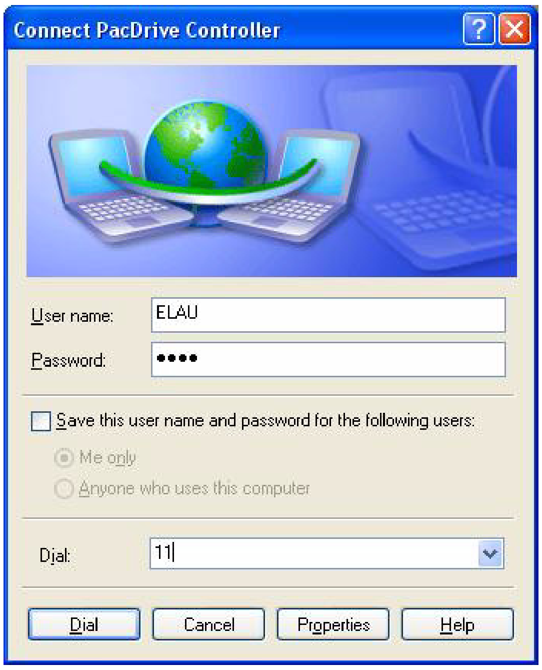
You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

[✓ Obtain an IP address automatically]
Use the following IP address:
IP address: . . .

[✓ Obtain DNS server address automatically]
Use the following DNS server addresses:
Preferred DNS server: . . .
Alternate DNS server: . . .

Bild 7-45: Properties of the internet protocol (TCP/IP)

- Click twice on the „OK“ button.



User name: ELAU

Password: ●●●●

☐ Save this user name and password for the following users:

Dial: 11

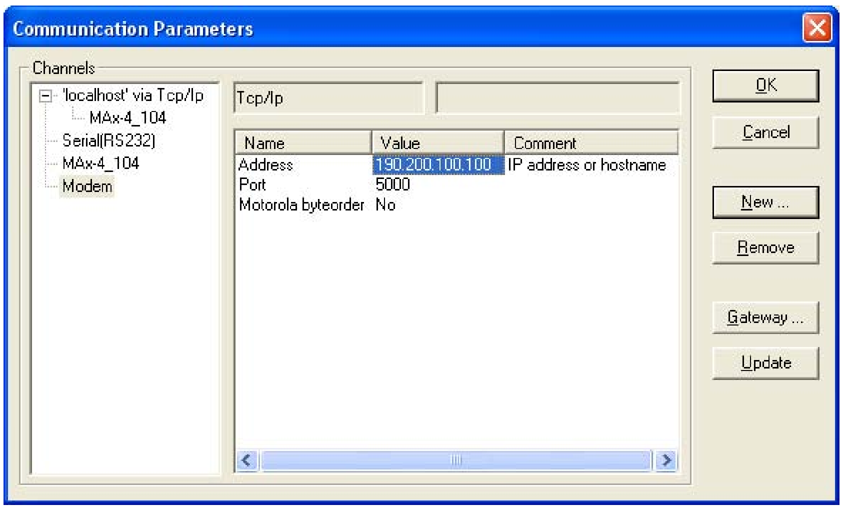
Fig. 7-46: Dialog to the connection setup

- Enter username (see parameter *UserName*) and password (see parameter *Password*) for the connection.
- Establish the connection with the button “Dial“.
- The connection is now available.

7.2.5 Establishing a Connection with Remote PC / PacController

The Operating Manual EPAS-4 includes a detailed description of how to establish a connection with the MAx-4 PacController. The following points are treated there:

- Checking the TCP/IP connection
- Setting up the gateway server
- Setting up the communication channel in EPAS-4



Name	Value	Comment
Address	190.200.100.100	IP address or hostname
Port	5000	
Motorola byteorder	No	

Bild 7-47: Communication parameter at modem connection

7.2.6 Trouble Shooting

NOTE

In principle, we recommend to make a test and carefully study the manual before commissioning.

Modem is dialling, but the modem on the MAx-4 PacDrive Controller does not take the call	
The modem on the PacDrive Controller is not or not correctly initialized. If CONTROL CONFIGURATION | MODEM | in the parameter *State* shows “offline“, the modem has been initialized. Check if the InitString configures the modem for taking calls	
Cause 1: Handling:	The parameter *Com1User* in EPAS-4 | CONTROL CONFIGURATION | GENERAL is not correct. Check the parameter *Com1User* . The parameter must be set to “Modem“ or “Auto“.
Cause 2: Handling:	The initialization string is not correct. In this case, the parameter *State* will show "[Exp]InitString ERROR" after the initialization. Check the initialization string in the parameter [*Exp*]*InitString* in the control configuration.
Cause 3: Handling:	The modem is not connected correctly or initialized incorrectly. In this case the parameter *State* will show "[Exp]InitString TIMEOUT". Check the power supply and the serial cable as well as the parameter [*Exp*]*InitString*.
Cause 4: Handling:	The parameter *ExpInitString* is not correct. Check the parameter *ExpInitString*.
Cause 5: Handling:	The chip set in your modem is no Rockwell chip set. In this case, the command for baud rate detection may not be recognized correctly. Use the valid commands for your modem, which are described in your modem manual. If there are no handshake problems between your modems, you can replace the content of the parameter *ExpInitString* by the command “AT“.
Cause 6: Handling:	The serial cable or the telephone cable does not match the modem on the PacDrive Controller. Use the cable provided by your manufacturer.

The modem ends the call before a connection has been established	
Cause 1: Handling:	The time for which the modem waits until a connection has been established was set too short. WIN9x : Telecommunication connection -> Properties -> Configure -> Settings : deactivate “Interrupt dialling“. WIN NT : Control panel -> Modem -> Properties -> Settings: deactivate “interrupt dialling".

The modem takes the call, but there is no handshake or the handshake is broken off	
Cause 1: Handling:	The modems are using different speed settings, i.e. one modem is set to “Connect with 19200 bps“, while the other modem only supports 14400 bps Set the modems to automatic speed selection. WIN9x : Telecommunication connection -> Properties -> Connect via -> Button “Configure“, register “General“: deactivate "Connect only with this speed" WIN NT : Control panel -> Modem -> Properties: deactivate "Connect only with this speed" If any, set the DIP switch on your modem correctly
Cause 2: Handling:	The modems do not recognize each other’s protocols Switch off the automatic modulation adjustment. To do this, add “+MS=,0“ to the parameter *ExpInitString* in the control configuration. Should your modem not support this command, please consult your manual.
Cause 3: Handling:	Poor telephone connection: It may help to switch the telephone provider or change the connection. Dial up the MAx-4 modem by telephone. Wait for the handshake signal of the modem (starts with a beep). If the signal is very low or there is a lot of background noise or crackling in the line, the quality of the line is probably too poor for a connection.

The connection has been established, but the telecommunication network shows “Verifying user name and password...“. After that the connection is cut off.	
Cause 1: Handling:	Password or user name are not identical with those saved in the MAx-4 PacDrive Controller (parameters *UserName* and *Password*) Enter the correct password and user name in EPAS-4.
Cause 2: Handling:	Only WinNT : The telecommunication network sent an encrypted password, which the MAx-4 does not recognize. Activate Telecommunication network -> button “More“ -> “Entry and modem settings...“ -> Security -> “Any authentification“.
Cause 3: Handling:	Telecommunication connection properties set incorrectly Win9x : under Telecommunication network -> button “Properties“ -> register “Server types“, deactivate everything except TCP/IP; Type of telecommunication server = "Internet, WIN NT Server, WIN9x" WIN NT : Telecommunication network -> button “More“ -> “Edit entry and modem settings“ -> register “Server“ -> Type of telecommunication server = "PPP, Internet, WIN NT Server, WIN9x" , activate TCP/IP and LCP extensions, deactivate all others
Cause 4: Handling:	TCP/IP protocol could not be started because IP addresses are invalid (e.g. reserved IP addresses were used). Use a valid IP addresses.
Cause 5: Handling:	The parameter *RemoteIpAddress* in Control configuration / Modem is not identical with the value in the telecommunication network. Check settings at WIN9x : Telecommunication network -> button “Properties“ -> register “Server types“ -> button “TCP/IP settings“. WIN NT : Telecommunication network -> Button “More“ -> “Edit entry and modem settings“ -> register “Server“ -> button “TCP/IP settings“.
Cause 6: Handling:	The modem settings are not identical Set “8 data bits, no parity, 1 stop bit“. WIN9x: Telecommunication network -> Properties -> register “General“ -> button “Configure“ -> register “Settings“: “8 data bits, no parity, 1 stop bit“. WIN NT : Control panel -> Modems -> Properties -> register “Settings“: set “8 data bits, no parity, 1 stop bit“. Same register (Fig. 7-14) -> button “Extended“: Data flow control: “Hardware (RTS / CTS)“ and “Error control“: Activate data compression. Should your modem have DIP switches, you may have to set them.

PD_UserMan_Teleservice_us.fm

When trying to log in, EPAS-4 reports: "Communication error: Logging out"	
Cause 1: Handling:	The communication parameter in the EPAS-4 menu ONLINE | COMMUNICATION PARAMETERS are not identical with *LocalIpAddress*. By default, *LocalIpAddress* is set to 190.200.100.100. Check the communication parameters
Cause 2: Handling:	The modem connection is too slow, e.g. less than 9600 BPS. EPAS-4 recognizes no connection because no complete data package is transmitted within the timeout limit. Raise the value of *EPAS_StandardTimeout* in the WINDOWS directory in EPAS-4.INI. After that, EPAS-4 must be restarted.
Cause 3: Handling:	There is already a device in the local network with the IP address chosen for the MAx-4 PacDrive Controller. Change the IP address of the PacDrive Controller. Disconnect the remote PC from the network. For questions about your IP addresses, please contact your system administrator.
Cause 4: Handling:	Network problem with unknown cause. Disconnect remote PC from the network and re-install telecommunication network from scratch.

EPAS-4 breaks off existing connection: "Communcation error: Logging out"	
Cause 1: Handling:	Modem connection remains active. The timeout limit for modem transmission has been exceeded. This happens in case of too low data through-put with slow modem connections and thus too long trans-mission times for data packages. Increase the value of **EPAS_StandardTimeout** in EPAS-4.INI in the WINDOWS directory to 10 or more. It might even be necessary to increase the value of **EPAS_ExtendedTimeout**. The worse the quality of the line and the slower the modem, the higher should you set this value.
Cause 2: Handling:	Modem disconnects: The connection was interrupted e.g. due to poor line qua-lity. Establish connection via a “better“ line.
Cause 3: Handling:	MAx-4 was rebooted. Wait for some seconds until the boot procedure is comple-ted and dial up again.
Cause 4: Handling:	Windows interrupted the connection because no data were sent via the telecommunication adapter for some time WIN9x : Telecommunication network -> Properties -> Con-figure -> Settings: deactivate “Disconnect after idle time " WIN NT: Control panel -> Modem -> Properties -> Set-tings: deactivate “Disconnect after idle time "
Cause 5: Handling:	Information of a foreign device (e.g. meter pulse, etc.) was sent to the modem. This caused the modem to interrupt the connection. Switch off any such additional function before establishing a connection. For information on this subject, consult the manual of your telephone provider.

7.3 Telecommunication network over Remote PC to the PacDrive Controller

7.3.1 Principle

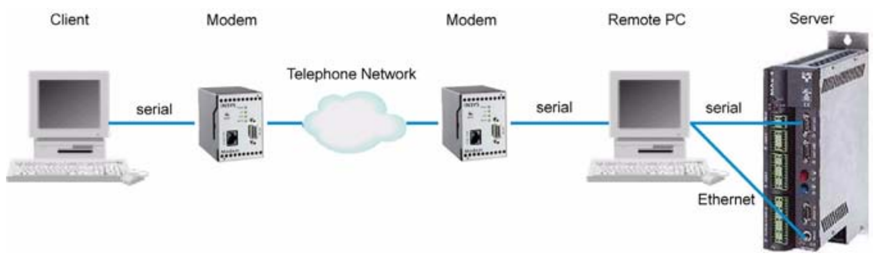


Bild 7-48: Direct telecommunication network over Remote PC to the PacDrive Controller

Advantages

- Connection to several PacDrive controllers possible
- Use of Windows “callback“ functionality possible
- Use of remote desktop connection or remote control tools such as VNC

Disadvantages

- More configuration work compared with a direct connection to the PacDrive controller
- Remote PC required on the PacDrive controller

7.4 Telecommunication Connection via Router to Several PacDrive Controllers

7.4.1 Principle

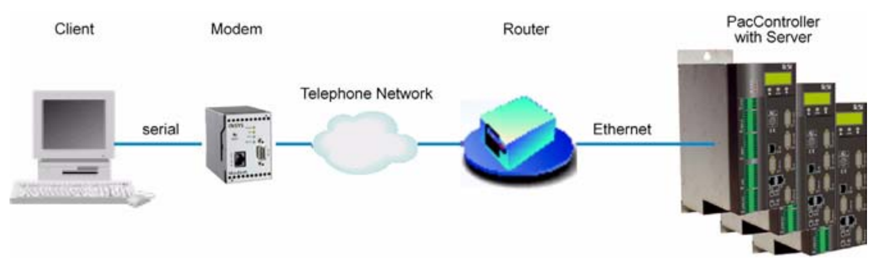


Bild 7-49: Direct telecommunication network over router to several PacDrive Controllers.

Advantages

- Connection to several PacDrive controllers possible

Disadvantages

- More configuration work compared with a direct connection to a PacDrive controller

7.5 Connection via Internet

7.5.1 Direct Connection via IP Address

To be able to establish a connection via Internet, you need to know the IP address of the PacDrive controller or remote PC. The TCP/IP protocol must be installed both on the client and on the server side.

However, the IP address cannot be chosen freely. A worldwide unambiguous IP address must be assigned.

This address is issued by a central international authority, the NIC (Network Information Center). Further information is available at www.denic.de.

Advantages

- Easy to configure

Disadvantages

- The possibility to apply for a worldwide unambiguous IP address for a PacDrive controller is probably irrelevant in practice (costs, limited number of IP addresses)

If, however, a remote PC dials up the Internet via a provider, that PC is automatically assigned a dynamic IP address. This makes it possible to establish a connection to that remote PC if that IP address is known.

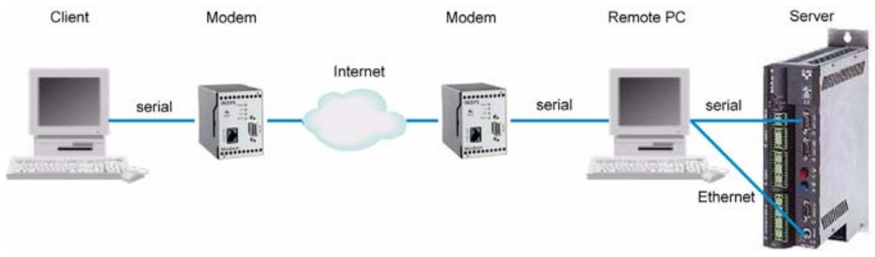


Bild 7-50: Direct connection over IP address and Remote PC

Advantages

- Connection with several PacDrive controllers possible
- Easy to configure

Disadvantages

- IP address changes with every dial-up

7.5.2 VPN (Virtual Private Network) Connection

With the help of a VPN connection, a network connection can be established via an Internet connection.

This is suitable e.g. if the PacDrive controller is integrated in an in-house network that has Internet access.

Advantage

- Behavior as if the connection were in a local area network (LAN)

Disadvantage

- Connection is complicated to configure

7.6 Access to the PacDrive Controller with a standard browser (not permited)

7.6.1 Prinzip

Teleservice via Internet with a “standard“ browser permits only very restricted access to the PacDrive™ M system functions.

WARNING!

Restricted view of the machine!

Risk of injuries and material damage!

- Only do teleservice if a qualified person is available on site, i.e. at the machine.
- Always keep in touch with this person (e.g. via telephone or video conference).

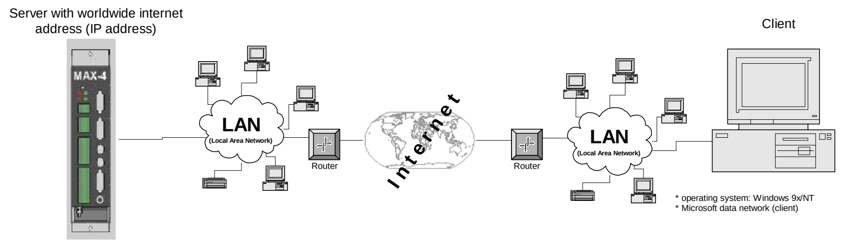

Server with worldwide internet address (IP address)
MAX-4
LAN
(Local Area Network)
Router
Internet
Router
LAN
(Local Area Network)
Client
* operating system: Windows 9x/NT
* Microsoft data network (client)

Fig. 7-51: Internet connection

Connection options

- Local area network (LAN)
- Telecommunication network via modem
- Connection to the Internet

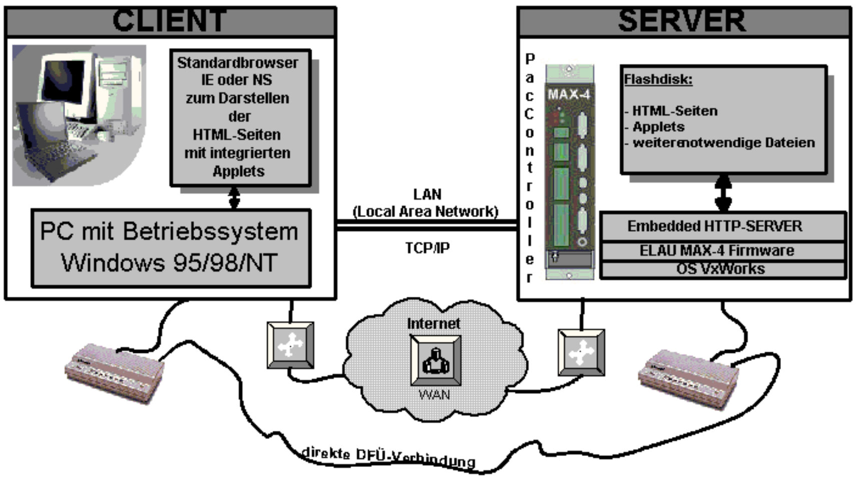


Bild 7-52: Komponenten der Ferndiagnose

The HTTP server running on the PacController makes it possible to load HTML pages with integrated diagnosis applets. The applets are implemented with various simple HTML tags. This guarantees that remote diagnosis is easy to handle and can be configured individually. You can determine design elements, contents and layout of an HTML page yourself. Then the designed page can be transmitted to the flash disk of the PacController with the help of an FTP client.

8 Networks

This chapter will give you an overview of the networking of devices.

You will be familiarized with goals, advantages, applications and technologies. You will know the transmission media and communication protocols, including the basics of TCP/IP. You will be shown tools for dealing with TCP/IP problems.

8.1 Basic Concept

In the Microsoft network environment, we need to differentiate between peer-to-peer environment or workgroup (not server-based) and domain (server-based).

8.1.1 Peer to Peer

All computer systems are equal and together form a workgroup.

Each computer provides resources and can access resources on other computers. To be granted access, a user must have a user account on each computer.

Positive No additional cost for server/operating system.

Negative No central user management; when the network reaches a certain size, the coordination becomes very complex.

Peer-to-peer is a cost-effective alternative for small networks with up to ten stations and without high security requirements.

Peer-to-peer operating system

Such operating systems are actually systems for stand-alone PCs that were extended by network functionality (Win95, Win98, WinME).

However, those systems lack significant features of “real“ network operating systems, e.g. log-in authentication. Their use is only sensible in the smallest networks arranged in the form of workgroups of otherwise independent PCs.

With Microsoft Windows XP a login is required.

8.1.2 Client-Server

With a client-server concept, usually a centralized user management is introduced. The users are broken down into different groups that have certain rights.

In a client-server concept the tasks are divided. One or several computers function as servers providing central resources and services. After successful login, all other computers can access those as clients.

In larger networks, different services are usually distributed on several specialized servers.

Typical server functions are: file server, print server, mail server, web server, proxy server.

Positive New users are quickly integrated thanks to the centralized user management.

The user can log into the network under his own name at any computer in the network and thus access the resources needed.

Negative Higher costs for additional server computers, server operating systems.

Client-server operating system

Operating systems providing centralized services that can be accessed by several clients, e.g. Win2000 Server.

Summary As existing peer-to-peer networks can be integrated in client-server networks, small companies can start out with peer-to-peer and later upgrade the network with a client-server structure, as needed. Both concepts do not exclude each other, but can be combined if and when needed.

8.2 Topologies

8.2.1 The Term “Topologies“

Difference between physical and logic topologies

Networking causes data traffic, and as with many other types of traffic, you can differentiate between traffic routes and traffic rules in the IT sector as well.

Physical Topology

The physical topology of a network refers to the traffic routes, the physical structure of the network is described. In simpler terms: The form in which the cables are laid in a wire-based communication system.

The most important basic forms of topology:

- Bus
- Star
- Loop

In practice, you will often find combinations of bus, star and loop.

Logic topology

The logic topology of a network describes the basic traffic rules on the communication routes, including who is allowed to access the transmission medium.

Relation

In practical implementation, the two terms are closely related, so that normally a certain physical topology entails a certain logic topology. However, physical and logic topology need not be identical.

8.2.2 Bus

All devices are connected to the same cable

The bus topology is characterized by a single central cable known as “bus“. All devices are connected to that bus and have to share that medium.

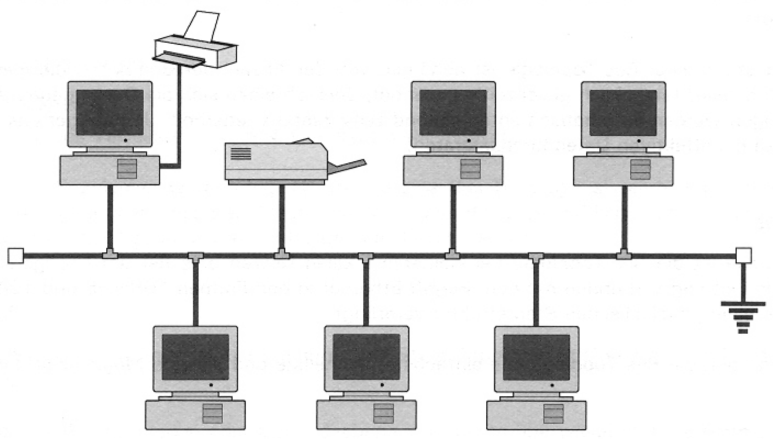


Fig. 8-1: Bus topology

Advantages of bus topology

- Relatively low costs, as it requires the smallest amount of cable.
- The failure of one station does not cause problems in the rest of the network.

Disadvantages of bus topology

- All data are transmitted via a single cable (one conductor).
- Data transmissions can be tapped quite easily.
- A distortion of the transmission medium at one place in the bus (defective cable, loose connector, defective network card) blocks the entire network.
- The search for the source of an error is often very tedious.
- Dense data traffic, as a station sends a message to all other stations.

8.2.3 Star

Every station has its own cable

In a star topology, every station is connected to a central distributor via its own cable.

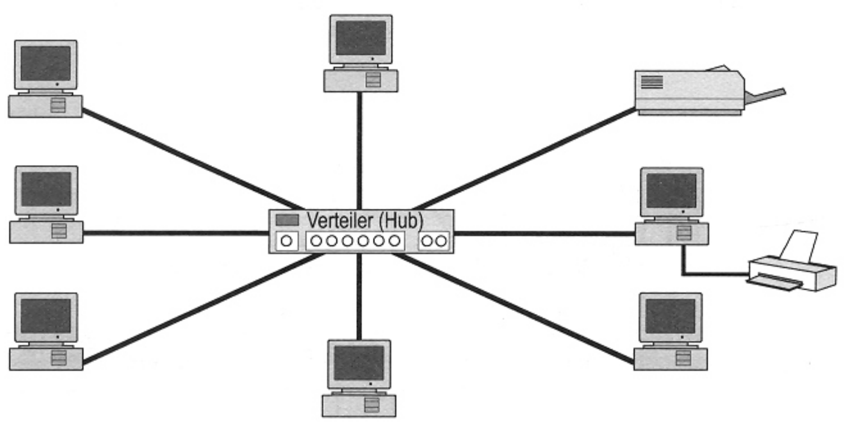


Fig. 8-2: Star topology

Advantages of star topology

- The failure of a station or defect of a cable has no effect on the rest of the network.
- Active distributors act as signal amplifiers.
- Given the functionality of the star distributor, individual computers need not share a cable for data transmission. This means that they can use the full bandwidth, so that this physical topology permits higher data throughput rates.
- Further stations and/or distributors can be added fairly easily.

Disadvantages of start topology

- Large amounts of cable
- If the distributor fails, network traffic is no longer possible.

8.2.4 Loop

Every station is connected to two neighbors

In a loop topology the cables form a closed loop. The cable has no beginning or end. All stations are added as elements to the loop, process and amplify the signals arriving on the cable and forward them.

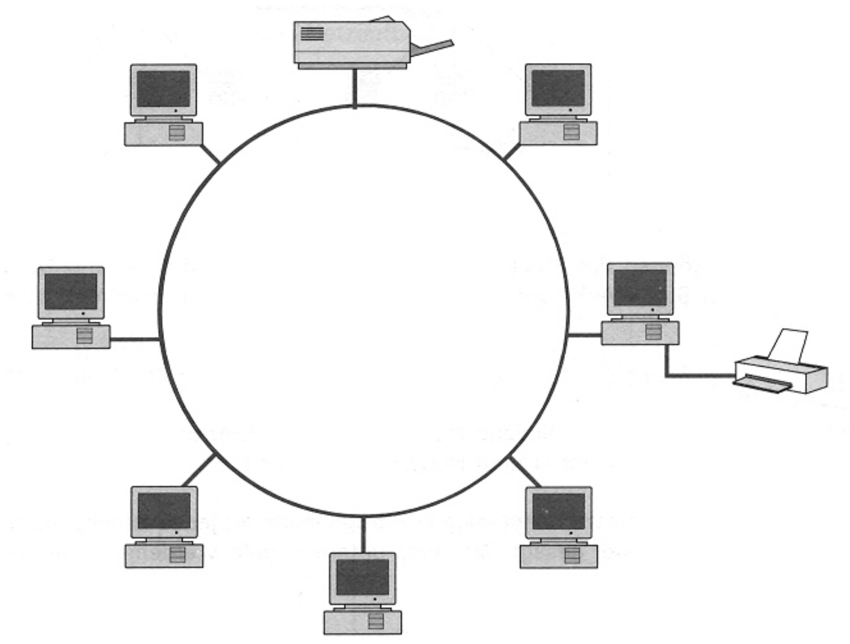


Fig. 8-3: Loop topology

This network concept is rarely used as physical topology, as the cabling is very complex.

The main disadvantage is that the failure of one station paralyzes the entire network.

8.2.5 Mixed Forms

Combinations of bus, star and loop

Of course it is also possible to develop mixed forms of the three basic forms. One reason for this may be that in growing corporate networks it becomes more and more likely that all the different typologies exist. Another reason is that a so-called backbone can be built up by mixing topologies.

Backbone A backbone is a physical connection of several partial networks, e.g. the connection of several buildings and their respective networks.

Star-bus network

Combined star-bus networks are common.

Several hubs form the centers of a star each, while the hubs are connected to each other by a bus cable.

A simple examplefor clarification: In a three-story building each floor has a star topology. The three floors, or rather the hubs, are connected to each other via one single bus cable.

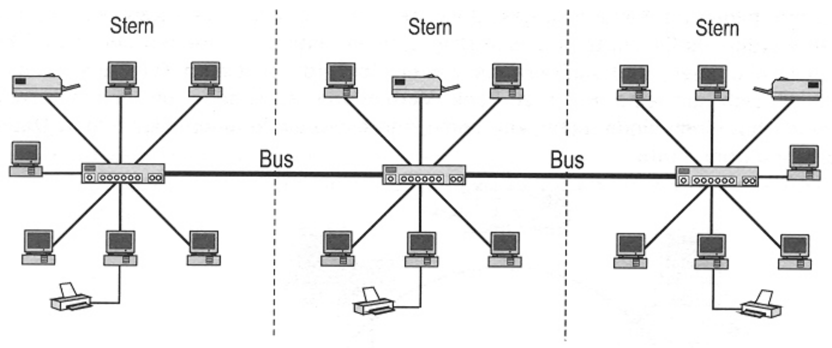


Fig. 8-4: Star-bus topology

If the bus cable fails, the floors can no longer communicate with each other. If a hub fails, the communication of the floors with each other and on the floor affected is interrupted.

Star-star network

Several hubs form the center of a star; the individual hubs are connected to a main hub.

Again, a simple example: In a three-story building each floor has a star topology. The three floors, or rather the hubs, are connected to the central hub via one cable each.

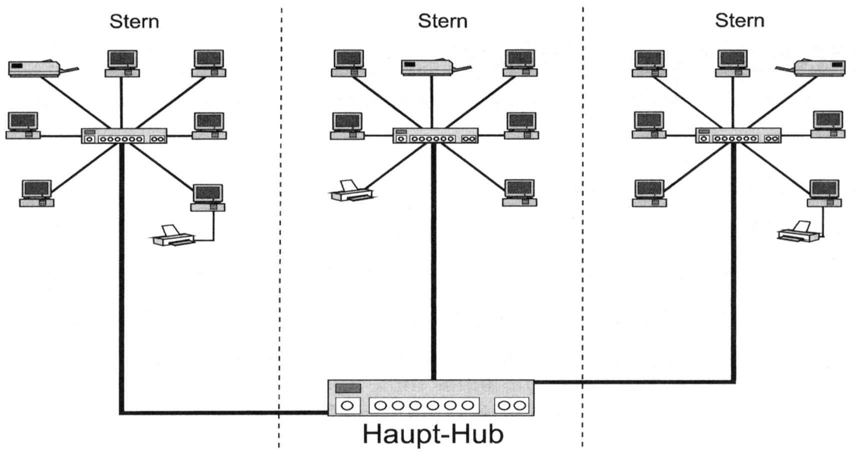


Fig. 8-5: Star-star topology

If the central hub fails, communication is only possible within the individual floors. If a cable from the central hub to a floor hub fails, that floor can no longer communicate with the other floors.

8.2.6 Transmission Media

Coaxial cable

Coax cables are available in different versions for different applications.

Nowadays they are rarely used for new cabling of computer networks.

The maximum transmission rate is 10Mbps.

Coaxes according to specification RG 58 are common.

RG-58 A/U - impedance 50 Ohm, thinnet 10Base2;

The thinnet cable is also known as “cheapernet“. In a bus network it is suitable for cable lengths up to 185 m. Networks with such cabling are called 10Base2.

The abbreviations 10Base2 and 10Base5 already include several characteristics of networking, namely the transmission speed of 10 Mbps in the base band and the extension limit of approx. 200 and 500 meters per segment, respectively.

With coaxial cables, the speed cannot be increased to 100 Mbps .

Twisted-pair cable

Cables with insulated leads twisted around each other. The twisting suppresses to a certain degree disturbance from outside or from neighboring pairs of leads.

Transmission rates of 100Mbps and more are possible. The maximum distance between the computer and a central component (e.g. hub) is usually 100m.

Unshielded twisted pair (UTP)

Cable with twisted pairs of leads without additional individual shielding.

- More susceptible to disturbing radiation!

Shielded twisted pair (STP)

Each pair of leads is shielded by a coat.

- Clearly less susceptible to electric disturbance.
- Data can be transmitted with higher transmission rates over long distances.

Category	Transmission rate	Application
1	1 Mbps	Analog language transmission, alarm systems
2	4 Mbps	IBM cabling type 3 (language), EIA-232
3	10 Mbps	10Base-T, 100Base-T4 4-MBit token loop, ISDN
4	16 Mbps	16-MBit token loop

Category	Transmission rate	Application
5 and 5e	100 Mbps	100Base-Tx, ATM (155 Mbps), SONET, SDH

Table 8-1: Network cable categories

Further categories (6 and 7) in the gigabit transmission range are in the process of being normed.

NOTE

If STP is used in a network, but one section uses UTP, the advantage of the STP cables is cancelled.

Fiber-optic cable

Signals are transmitted unidirectionally via light impulses, i.e. only in one direction, which is why at least two fibers (lanes) per cable are needed. The light impulses are transmitted by means of a laser diode or LED.

- High transmission rates (gigabit range) and long distances
- Good safety (against disturbing radiation and taping)

There are two different modes of fiber-optic cables:

Monomode

The light runs practically parallel in the glass fiber core, so that light reflections (dispersion) are minimal.

- 100GHz x km

Multimode

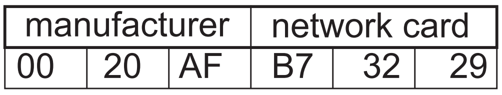
The core is thicker than the monomode fiber, dispersion increases.

- 1GHz x km

8.3 MAC Address

A worldwide unique address is burnt into every network card.

Standardized structure of a MAC (media access control) address.



manufacturer			network card		
00	20	AF	B7	32	29

Fig. 8-6: Structure of the MAC address

The MAC address is shown by entering *ipconfig /all* or *GetMac* at the input prompt.

8.4 Access Methods

8.4.1 CSMA/CD

The access method of the Ethernet is CSMA/CD (carrier sense multiple access with collision detection).

Each station can start sending whenever it wants. The individual stations have competing access to the common transmission medium at any time (multiple access).

Sending procedure:

When a station is sending, the signals are transmitted via the cable in both directions. By means of regulation you try to minimize the risk that two stations start sending at the same time and thus the signals are destroyed on their way (collision). Regulation means that, before sending, a station checks the transmission medium to see if it is free. If there are already signals in the cable, the station will not start sending in the first place.

Collision detection:

If there is a collision anyway because two stations find the cable “free“ and start sending at the same time, that collision must be detected and a reaction must be triggered.

To detect a collision, all stations continue monitoring the transmission medium. The station that first detects a collision sends a so-called JAM signal. Each station that registers the JAM signal stops sending data immediately. When the line is free again, new attempts to send can be started. Either a new attempt to send is started after an arbitrary delay (non persistent CSMA), or the station that wants to send continues monitoring the medium and starts sending.

as soon as it finds the medium free (1-persistent CSMA).

- The more stations are connected and trying to send, the more often do collisions occur, and the lower is the effective data throughput.
- The time of sending cannot be calculated or determined, but is arbitrary.
- The method is therefore unsuitable for time critical applications.

8.4.2 Token Passing

Access method for token-loop networks (e.g. ArcNet).

Each client in the token loop is allowed to send in a given and thus calculable time frame. This method is deterministic.

Sending method:

When a station wants to send, it has to wait until it receives the token circulating in the loop (a bit pattern circulating in the network). Within the bit pattern of a token, there is a digit that shows if the token is actually free or busy.

When a station receives a “free“ token, it can change it into “busy“ and add information on the source and target addresses as well as the data to the token. The whole package is now called a frame and forwarded to the next station. That station checks the receiver address and, if it is not the addressee, forwards the message.

Receiving

When the message has reached its destination, it is copied there. The destination acknowledges receipt by an entry in the frame and forwards the modified package to the next station.

When this (acknowledged) package finally arrives at the former sender station, that station recognizes that the message has been transmitted. Now the station replaces the frame by a free token, which it forwards to the next station.

Access rules

Apart from this basic structure, token loop has further rules to ensure data transmission. In each loop one station takes the role of the “active monitor“. Usually it is the first station that becomes active in a loop. When that computer is switched off, a new station for that role is determined in a defined procedure (token claiming procedure). The active monitor ensures, among other things, that there is always a valid token or a valid frame (token plus data) in the loop.

- The network can be extended up to the maximum permitted number of stations without impairing the regulated data throughput.
- With the NAUN procedure, token passing can find out relatively quickly if a station is causing a problem, and which station in the loop that is. The station can automatically be removed from the loop.
- Suitable for time-critical applications.

PD_UserMan_Netze_us.fm

8.5 Overview of Network Operating Systems

8.5.1 Novell NetWare

The proprietary IPX/SPX protocol is used. As of version 4.1, however, the system also supports TCP/IP, which became the standard protocol with version 5.0.

8.5.2 Microsoft Windows NT and Windows 2000

Domain concept in NT

The directory services in Windows NT are implemented in the so-called domain concept. A domain is an administrative unit which is eventually created by a single server, the so-called primary domain controller (PDC).

Administration is done via (user) nodes stored in a user node database on the PDC.

Administration of that database is also possible from other computers.

Active Directory – further development of the domain concept in Windows 2000.

Windows 2000 integrates the domain concept into ADS (active directory services).

This makes the creation and administration of large networks significantly easier than in Windows NT. Existing NT 4.0 domains can be integrated in the active directory fairly easy.

8.6 TCP/IP

8.6.1 Comparison of Reference Models

TCP/IP proceeds from a four-layer architecture model and can be reconciled with the seven- or eight-layer OSI reference model:

Layer	OSI	TCP/IP	Layer
7	Application layer	Application layer	4
6	Presentation layer		
5	Session layer		
4	Transport layer	Transport layer (TCP)	3

Layer	OSI	TCP/IP	Layer
3	Network layer	Internet layer (IP)	2
2	Data link layer	Network layer	1
1	Physical layer		

Table 8-2: Comparison: OSI <-> TCP/IP

8.6.2 Important Protocols

In addition to the TCP and IP protocols, which give the protocol family its name, there are a variety of other protocols (>100)

Protocol	Name	Description
TCP	Transmission Control Protocol	For establishing logic connections between applications. It is located on the transport layer and, as a connection-oriented protocol, serves for secure data transmission.
UDP	User Datagram Protocol	Also located on the transport layer, but working without connection. Therefore it is not as secure as TCP, but faster.
IP	Internet Protocol	A connection-fee protocol, serves for packaging and package switching via IP addresses.
IPSec	IP Secure	Extends the regular IP protocol by a number of security mechanisms.
FTP	File Transfer Protocol	Serves for data exchange between computers and is located on application level.
HTTP	Hypertext Transfer Protocol	Serves for the transport of HTML (Hypertext Markup Language) pages and is located on application level.
Telnet	Telecommunication Network Protocol	Terminal emulation for host communication. Telnet is located on application level.
SMTP	Simple Mail Transfer Protocol	For sending e-mails, located on application level.
POP	Post Office Protocol	For receiving e-mails, located on user level.

Protocol	Name	Description
ARP	Address Resolution Protocol	For resolving logic IP addresses into physical MAC addresses.

Table 8-3: Overview of important protocols

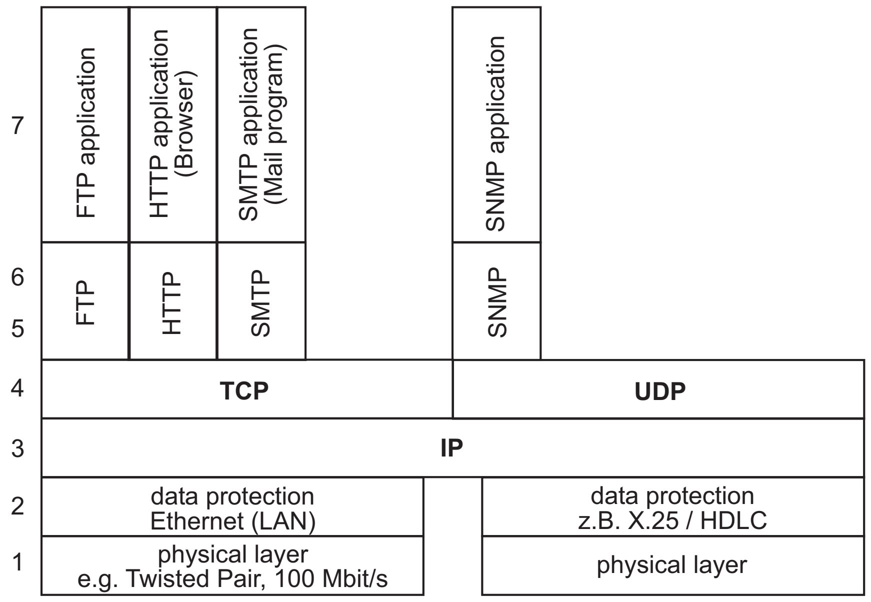

7
6
5
4
3
2
1
FTP application
HTTP application (Browser)
SMTP application (Mail program)
SNMP application
FTP
HTTP
SMTP
SNMP
TCP
UDP
IP
data protection Ethernet (LAN)
data protection z.B. X.25 / HDLC
physical layer e.g. Twisted Pair, 100 Mbit/s
physical layer

Fig. 8-7: Structure of protocols and services and their location in OSI layers

Port number:

As different services such as FTP, Telnet, SMTP or HTTP can be called at the same IP address in a computer that can be reached by TCP, an additional marker is needed to make it clear which of those services is requested.

This is done by means of the so-called port number.

Number	Meaning
20	FTP data
21	FTP commands
23	Telnet
25	SMTP
80	HTTP
110	POP3

Table 8-4: Port numbers

PD_UserMan_Netze_us.fm

Socket:

Whereas the IP address unmistakably identifies a computer, the port number determines which service on the computer is addressed.

Both pieces of information together are called socket.

8.6.3 IP Addressing

When using TCP/IP, an unambiguous IP address must be assigned to each network card within a network. Thus logic IP addresses are assigned to the physical network card addresses (MAC addresses).

For computers that are connected to the Internet, this means that their IP addresses are worldwide unique.

8.6.4 Subnet Mask

The subnet mask must be entered when installing the network protocol.

When a computer in the network receives a data package, if first checks with the help of the subnet mask if the package is at all meant for a computer in the network segment. If there is a match, it then checks if the node number is identical to its own. If this is also the case, it reads the entire contents of the data package.

Example

	Network ID			Host ID
IP address - decimal - binary	192 1100 0000	168 1010 1000	10 0000 1010	1 0000 0001
Subnet mask - decimal - binary	255 1111 1111	255 1111 1111	255 1111 1111	0 0000 0000

Table 8-5: Correlation between IP address and subnet mask

PD_UserMan_Netze_us.fm

8.6.5 IP Address Classes

The central authority for the assignment of IP addresses is called InterNIC (Internet Network Information Center).

In Germany, the office in charge is DENIC (Deutsches Netzwerk-Informations-Center).

However, those centers do not issue individual IP addresses, but only address groups, which providers then pass on to their customers. Providers can request different numbers of IP addresses. This is why IP addresses were broken down into different address classes.

Classes A, B and C are for addressing computers.

Class	Network ID	Subnet mask	Number of networks	Number of network nodes
A	0 to 126	255.0.0.0	126	16.777.216
B	128.0 to 191.255	255.255.0.0	16.384	65.536
C	192.0.0 to 223.255.255	255.255.255 .0	2.097.152	256

Table 8-6: Network address classes

Addresses starting with 127 must not be assigned, as they are reserved for special functions, e.g. 127.0.0.1, which usually addresses the so-called localhost, the internal loopback address.

8.6.6 Private IP Networks

For the creation of private networks, including many small LANs, one section each was excluded from the network classes described above.

The addresses in those three areas are <u>never assigned as public Internet addresses</u> and generally not forwarded by routers on the Internet.

10.0.0.0	to 10.255.255.255	in class A
172.16.0.0	to 172.31.255.255	in class B
192.168.0.0	to 192.168.255.255	in class C

If, however, a LAN is to be addressed via Internet, it must be guaranteed that the IP address is unambiguous in the entire Internet.

In this case, a <u>public IP address is needed</u>.

8.6.7 IP Address Assignment

Windows 2000 offers four methods for the assignment of IP addresses to TCP/IP clients:

- DHCP
- APIPA
- Static IP addressing
- Alternative IP configuration (multiple networking)

DHCP

DHCP permits the automatic configuration of IP addresses and other configuration options for clients in a network with one or several DHCP servers.

So-called IP segments (IP addresses to be assigned) are created on the DHCP server and configured with options such as standard gateway or DNS server. When booting, a client asks the DHCP server for an IP configuration, which it can then use for a certain time (lease time). When shutting down, the client logs off the DHCP server, which then assigns the IP address to another client.

APIPA

APIPA is suitable for simple networks with only one subnet. If no DHCP server is available, the computer automatically assigns itself a private IP address via APIPA.(169.254.0.1 to 169.254.255.254)

Later on, when a DHCP server is available and determines an address, the computer will change its IP address accordingly.

Static IP addressing

Static IP addressing permits the manual configuration of IP addresses if DHCP and APIPA are not available or technically impossible.

Particularly in large networks, this method is very time-consuming and prone to errors.

Alternative IP configuration (multiple networking)

With the alternative IP configuration, several addresses can be assigned to a network card. If connections need to be established with different networks (e.g. different locations), one IP address from each of the respective subnets is assigned to the network card.

8.6.8 Name Resolution

A computer name is resolved into the corresponding IP address.

Windows 2000/XP offers four different methods for resolving names in IP addresses.

This includes DNS, WINS, name resolution with HOSTS or LMHOSTS files and broadcast name resolution. Usually Windows 2000/XP uses a combination of the methods.

DNS

Queries are sent to DNS servers. The method is used for applications and services requiring host-to-IP name resolution. This method is needed if the computer wants to access the Internet.

DNS is the standard method for name resolution for clients in Windows XP.

When the resolution service of Windows XP receives a positive or negative reply to the query, it inserts that reply into its cache and thus creates a DNS resource entry. Before the query to DNS servers, the resolution service always checks the cache first. If there is a DNS resource entry in the cache, the resolution service uses the entry saved there and executes no server query. This reduces the queries and accelerates the network traffic due to DNS queries.

Show DNS resolution cache:

```
ipconfig /displaydns
```

Manually clear DNS resolution cache:

```
Ipconfig /flushdns
```

NetBIOS name resolution

Queries are sent to WINS servers. This method was added for compatibility reasons for applications and services requiring NetBIOS-to-IP name resolution, e.g. the search function of Windows NT 4.0, Windows 98/95.

8.6.9 IP Host and NetBIOS Name Resolution

Executed by HOSTS or LMHOSTS files.

Via manually managed local files, they provide host-to-IP and NetBIOS-to-IP name resolution.

NetBIOS name resolution (broadcast)

Executed by means of B-node broadcast. With the broadcast, the name resolution is executed within the local subnet.

8.7 Coupling of Networks

8.7.1 Additional Devices for Network Coupling

To enlarge networks, additional devices are needed to overcome existing limitations.

Only some of the devices are able to connect several segments (parts of an individual network). Others can couple several independent networks (subnets).

Repeater

A repeater (signal amplifier) is an active network component that takes over a regeneration functionality.

A repeater receives signals, amplifies and forwards them.

In local networks, a repeater is used to increase the maximum signal range and thus the maximum possible cable length. A repeater can thus connect network segments of the same LAN type for communication purposes.

A repeater works on layer 1 of the OSI model, i.e. the repeater only regenerates bit streams that are sent via the media, but has no insight into the messages. Therefore the same access method to the medium must be used in all segments involved. A repeater cannot mediate between e.g. Ethernet and token loop.

As a layer 1 device, a repeater is completely “transparent“, i.e. neither other connected devices nor programs “know“ about the device.

Repeaters no longer play any major role in new networks.

HUB

HUBs, like repeaters, work on layer 1 of the OSI model and are completely transparent to other systems.

By connecting several hubs, the number of connected stations can be increased fairly easily.

HUBs are connected to each other via a crossed cable - or if a straight cable is to be used, a special input (uplink input) must be used on the HUB.

The respective terminals are each connected to the HUB with a cable of their own (straight). Such a connection point is known as port.

As each connected device has its own decided connection and its own cable to the HUB, HUBs can offer those devices the full bandwidth of the local network at least up to the HUB port. Nevertheless, the entire segment remains, i.e. the devices have to share the transmission medium - every node receives every message - even messages meant for other stations.

Bridge

A bridge already operates on layer 2 of the OSI model (MAC address level) and thus is transparent to higher protocol layers. This means that all protocols based on layer 2 are forwarded by the bridge without being processed. In contrast to the repeater, a bridge can work with different transmission rates and different access methods and thus can also be used e.g. between CSMA/CD and token passing. A bridge connects network segments which, in contrast to the repeater, can also represent different technologies (e.g. Ethernet and token loop). If the bridge knows the recipient, it will forward the data only to the appropriate network.

A bridge cannot narrow down circular messages in the form of broadcasts to one segment, i.e. the broadcasts from one computer to all others are forwarded to the other segments.

A bridge temporarily saves and prepares the data packages before forwarding them on the basis of the MAC address. For this task, a bridge uses address tables (bridge tables) in which the MAC addresses of the connected stations are entered.

Switch

A switch is primarily the further development of a bridge. Just like a bridge, a switch works on layer 2 of the OSI model and uses MAC addresses to deliver the data.

A bridge can improve the bandwidth in the entire network by forwarding data traffic only if it is a transmission from one segment to another.

A switch, on the other hand, can improve the bandwidth within a segment.

In contrast to the bridge, direct and decided connections are switched between the ports. Data packages are sent to the port of the target station only. This reduces the network load substantially.

As the switch works on the MAC address level, a switch cannot couple two network segments.

Several manufacturers are already offering high-capacity switches that work not only on layer 2, but up to layer 3 or even higher.

Layer 3 switch / layer 4 switch

Whereas a layer 2 switch only has the MAC address available to forward the packages and a layer 3 switch additionally uses the target IP address, a layer 4switch can also steer the data traffic by means of the port number.

Router

A router is a device that can couple separate networks or divide large networks into subnets.

The simplest form of a router is a PC with several network cards, each of which has contact to different networks.

Routers work on layer 3 of the OSI model. This means that routers can connect networks with different topologies and the layers 1 and 2 lying beneath them.

However, all networks involved always have to use the same way of addressing their data packages.

The router unpacks arriving data packages up to layer 3 in order to find out the target address in the header. This procedure takes time, so that routers are usually slower than switches.

The allocation of routers to OSI layer 3 also means that a router does not use the MAC address, but the address of the protocol level (e.g. the IP address) as a target.

Routing tables

The main function of a router is to find routes for packages from source to target.

For this purpose, routers use so-called routing tables, where known target addresses are listed.

When a package arrives and the router has read the desired target address, it can check the routing table to see if that address is registered there.

If this is the case, it can forward the package. If this is not the case, it forwards the package to another router (default router/gateway), which then tries in its own way to deliver the package. This forwarding is also called “hop“.

This means that the gateway to be entered in the Windows control panel is actually a router.

Routing with Windows 2000

On each Windows 2000/XP computer, the static routing table of the adjacent segments is created automatically. For this purpose, the system evaluates the actual configuration information of the IP protocol for each network card that is installed and configured.The service Routing and RAS must have been started (see PacDrive P and Ethernet).

To test the routing properties of a Windows2000/XP computer, there are various tools, which we will deal with later on.

Gateway

A gateway connects networks with completely different protocols and address formats.

A gateway can connect actually incompatible networks. This is made possible because a gateway can be active on all 7 OSI layers. In the extreme, it unpacks an incoming message up to layer 7 in order to re-pack it down to layer 1 in a way that is suitable for the other network.

A gateway really converts one protocol into another.

8.7.2 Tools for Dealing with TCP/IP Problems

Ping (ping IP address)

Ping is a tool to check the connectivity on IP level. With the command ping, an ICMP echo request is sent to a target host. You should always use ping when you want to check if a host computer can sent IP packages to a target host.

```
Ping IP address of the remote host
Ping host name of the remote host
```

Ping uses name resolution to resolve a computer name into an IP address. If the ping signal can be sent to an IP address successfully, but sending the signal to a name fails, this is due to the host name resolution, not the network connectivity.

If the ping cannot be used successfully at all, make sure that the following prerequisites are fulfilled:

- The IP address of the local computer is valid and is shown properly on the GENERAL tab in the dialog field Internet protocol properties (TCP/IP) or when using IPCONFIG.
- A standard gateway has been configured and the connection between gateway and host is ready. Although several standard gateways can be configured from Windows 2000, the gateways after the first one are only used if the IP stack detects that the original gateway is not working. In this case all other gateways should be deleted to make things easier.

NOTE

If there is a connection with a long delay to the remote system to which the ping signal is sent, the reply may take longer. The default time limit of four seconds can be extended with the parameter – w (wait).

Frequent error messages returned by ping:

Error message	Meaning and action
Time to live exceeded during transmission	The number of required hops exceed the time to live (TTL). Extend the TLL by means of the parameter *ping -i*.
Target host cannot be reached	The sending host or a router has no local or remote route for the target host. Update the routing table on the local host or router (*route add*).
Request time excee-ded	No messages for the echo reply received within the given time (standard setting: 4 seconds). Extend the time by means of the parameter *ping -w*.
Ping request could not find host	The name of the target host cannot be resol-ved. Check the name and availability of DNS or WINS servers

Table 8-7: Meaning and action in case of error messages with “Ping“ command

Deleting the ARP cache

If a ping signal can be sent successfully both to the loopback address (127.0.0.1) and to the own IP address, but not to another IP address, the ARP cache may have to be deleted.

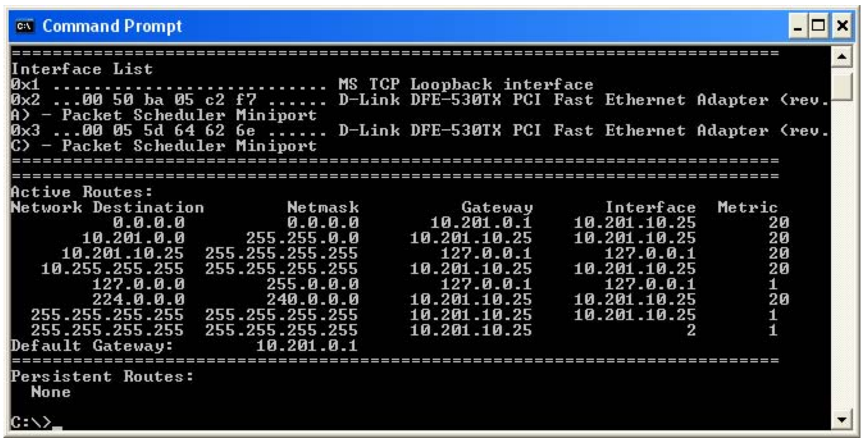
The tool ARP is used for that purpose. The cache contents can be displayed with the commands arp –a and arp -g. The entries are deleted with the command *arp –d IP address*.

The complete ARP cache is deleted with the command *netsh inter-face ip delete arpcache*.

Route print

The routing table is shown on the computer with the command “route print“.

For two hosts to be able to exchange IP datagrams, they must have a route to each other or use a standard gateway that knows a route.



```
===========================================================================
Interface List
0x1 ........................... MS TCP Loopback interface
0x2 ...00 50 ba 05 c2 f7 ...... D-Link DFE-530TX PCI Fast Ethernet Adapter (rev.
A) - Packet Scheduler Miniport
0x3 ...00 05 5d 64 62 6e ...... D-Link DFE-530TX PCI Fast Ethernet Adapter (rev.
C) - Packet Scheduler Miniport
===========================================================================
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0       10.201.0.1    10.201.10.25       20
       10.201.0.0      255.255.0.0     10.201.10.25    10.201.10.25       20
     10.201.10.25  255.255.255.255        127.0.0.1       127.0.0.1       20
   10.255.255.255  255.255.255.255     10.201.10.25    10.201.10.25       20
        127.0.0.0        255.0.0.0        127.0.0.1       127.0.0.1       1
        224.0.0.0        240.0.0.0     10.201.10.25    10.201.10.25       20
  255.255.255.255  255.255.255.255     10.201.10.25    10.201.10.25       1
  255.255.255.255  255.255.255.255     10.201.10.25               2       1
Default Gateway:        10.201.0.1
===========================================================================
Persistent Routes:
  None

C:\>_
```

Fig. 8-8: The command “route print“ at the MS-DOS prompt

The routing table shows a computer with IP address 10.201.17.124, subnet mask 255.255.0.0 and standard gateway 10.201.0.1. It contains the following entries:

Network target

The network address in the routing table is the target address. The column for the network target can contain three different types of entries, which are listed below in the order of their accuracy.

1) Host address (a route to a single, specific target IP address)

2) Subnet address (a route to a subnet)

3) Standard route 0.0.0.0 (This route is used if no other match could be found)

4) If no match is found at all, the packages are deleted.

<u>The entries of the routing table in detail:</u>

1) The standard route used for forwarding packages to all locations outside the local subnet.

2) A subnet route for the local subnet.

3) A host route for the local host (route for packages sent to that host computer).

4) A host route for a special IP broadcast address, the broadcast address for all subnets.

5) The loopback route a host uses when sending packages to its own address.

6) The IP multicast route the computer uses when sending packages to an IP multicast group.

PD_UserMan_Netze_us.fm

7) A host route for the limited broadcast address.

Network mask The network mask indicates which part of the target address must match the network address so that a route is regarded as a match with the target address of the forwarded package. If the mask is given in binary format with ones and zeros, the value 1 stands for a bit that has to match and 0 for a bit that does not have to match.

In a mask that consists only of ones (255.255. 255. 255) the target address of the package to be forwarded must exactly match the entire host address for this route in order to be regarded as a match. In the example of the network mas 255.255.0.0 the first two octets must match exactly. The last two octets need not match.

Gateway address The IP address for forwarding is determined on the basis of the gateway address in the routing table. It is either the host’s own IP address or that of a router in the local subnet. If the gateway address of the route is the host IP address, the IP address for forwarding is set to the target IP address in the IP datagram. This is the IP address of the host to be contacted. It is insignificant whether the host is located in the local subnet or in a remote subnet. If the gateway address of the route is not the host IP address, the gateway address, usually the address of a router in the local subnet, is used as the IP address for forwarding.

Route add

If a route is missing between two hosts that can be connected, a route can be added with the command “route add“.

Example:

```
ROUTE ADD 192.168.3.0 MASK 255.255.255.0 192.168.2.2
```

The MASK option routes entire networks; i.e. if you want to address a host in the network 192.168.3, then use router 192.168.2.2

Tracert

With the traceroute command (tracert) a route to a certain given IP target can be traced via up to 30 hops (router/gateway).

The speed of data transmission on the respective sections is protocolled as well.

E.g. tracert 10.205.0.1 (shows the package route to the target 10.205.0.1)

Pathping

The command pathping sends data packages to each router on the way to the last network target IP address and then reports via the path of the packages from one router to the next.

As pathping indicates the extent of package loss in route segments or connections, it can be detected exactly which routers or connections may be overloaded and are causing network problems.

Connection order

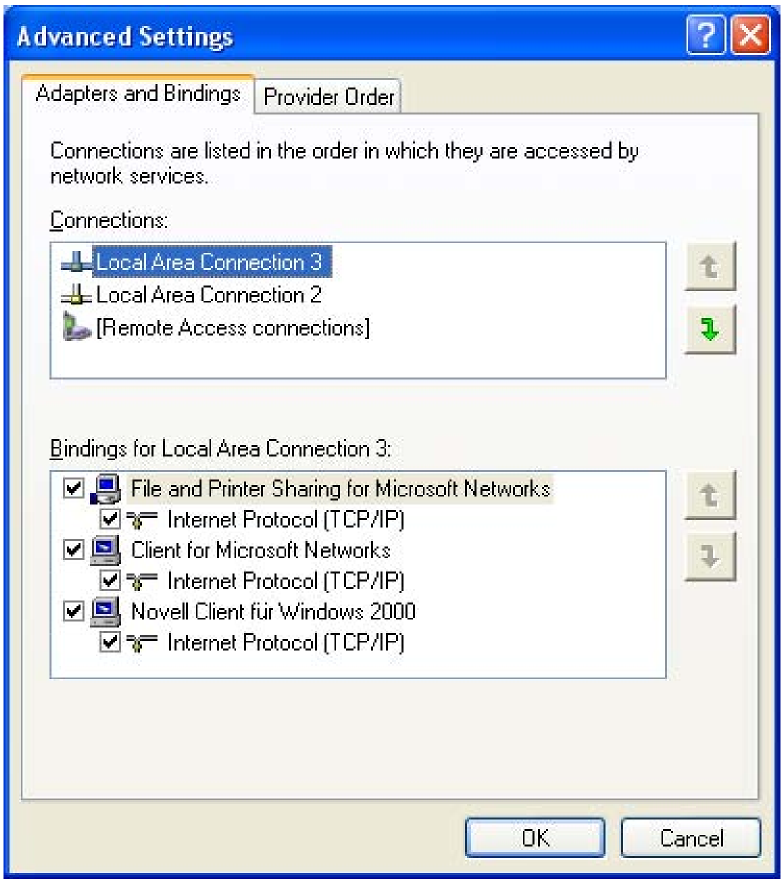
If several network cards and several network protocols are installed on a computer in Windows XP Professional, you can define the connection order of the respective network cards and protocols for the services using the respective protocols.

The connection order defines which protocol a service uses for the connection to another service or client and which network card is used for the connection.

To reduce the time for searching the necessary clients and services, place the most frequently used network card and protocol first.

Many services can establish connections with any protocol.

Network access, however, is controlled by the service Client for Microsoft Networks. The connection order is shown on the tab Network cards and connections on the properties page Extended settings (right mouse button - Network environment | Advanced | Advanced settings) of a selected network adapter.



Advanced Settings

Adapters and Bindings | Provider Order

Connections are listed in the order in which they are accessed by network services.

Connections:

- Local Area Connection 3
- Local Area Connection 2
- [Remote Access connections]

Bindings for Local Area Connection 3:

- ☑ File and Printer Sharing for Microsoft Networks
  - ☑ Internet Protocol (TCP/IP)
- ☑ Client for Microsoft Networks
  - ☑ Internet Protocol (TCP/IP)
- ☑ Novell Client für Windows 2000
  - ☑ Internet Protocol (TCP/IP)

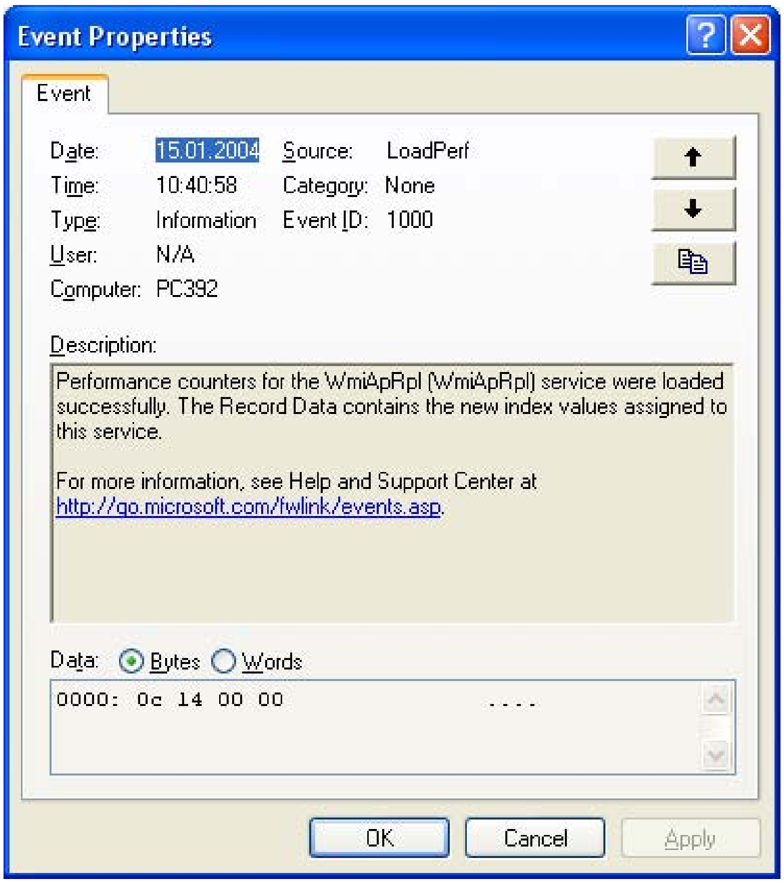
OK | Cancel

Fig. 8-9: Configuring the order of connection for the network card and the order of protocol creation

Display of events

Window order:

Workplace | Settings | System programs | Events | Safety



Date: 15.01.2004 Source: LoadPerf
Time: 10:40:58 Category: None
Type: Information Event ID: 1000
User: N/A
Computer: PC392

Description:

Performance counters for the WmiApRpl (WmiApRpl) service were loaded successfully. The Record Data contains the new index values assigned to this service.

For more information, see Help and Support Center at http://go.microsoft.com/fwlink/events.asp.

Data: [✓ Bytes] Words

```
0000: 0c 14 00 00                      ....
```

Fig. 8-10: Dialog window “Properties of Event“

The display of events enables you to monitor events in the system. Windows XP starts the event protocol service by default.

A typical protocol entry is composed of the header and a description.

9 System Data

Size	Value
IEC tasks	max. 64
IEC program code (code size)	2 MByte
IEC data area (data size)	2 MByte (in versions < V00.07.00 1 MByte)
IEC variable area markers inputs outputs	data size - 3 x 64 kByte 64 kByte 64 kByte 64 kByte
IEC retain variable	
NvRam MAx-4 - size - data retention	 1 kByte max. 7 days
flash Disk MAx-4 with UPS object - size - data retention - write cycles	 64 kByte > 100 years max. 100 000
NvRam with PN-4 module - size - data retention - battery buffer	 101 kByte max. 7 days (without battery) can be done internally or externally
Maximum variable sizes	
Array	128 kByte (32 kByte < V00.12.00)
Struct	128 kByte (32 kByte < V00.12.00)
global variable list	128 kByte (32 kByte < V00.12.00)
size of all variables of a POU of the type FB (including POUs called up)	128 kByte (32 kByte < V00.12.00)
stack depth for functions	10 kByte
nesting depth if instruction	15
SERCOS cycle time	
accuracy	< 0.01 %
resolution	1, 2, 4 ms (can be set with parameter *CycleTime*)
System clock	
accuracy	< 0.1 %
resolution	250 µs

Size	Value
real-time clock on the PN-4 module - accuracy - data retencion - battery buffer	 ±1 minute in 12 days. max. 7 days can be done internally or externally

Table 9-1: Data of the PacDrive™ M system

NOTE

Bit variables declared with the keyword AT can only be placed on the first 4096 bytes. Otherwise there will be a syntax error in the declaration.
The initialization behavior can be influenced with the object parameter MAx-4.General.RetainInit.
After successful compilation of a project, the sizes *Code Size* and *Data Size* are shown.

10 Time Diagrams - MotorController

10.1 Switching on Enable

Switching on enable results in the following behavior over the time:

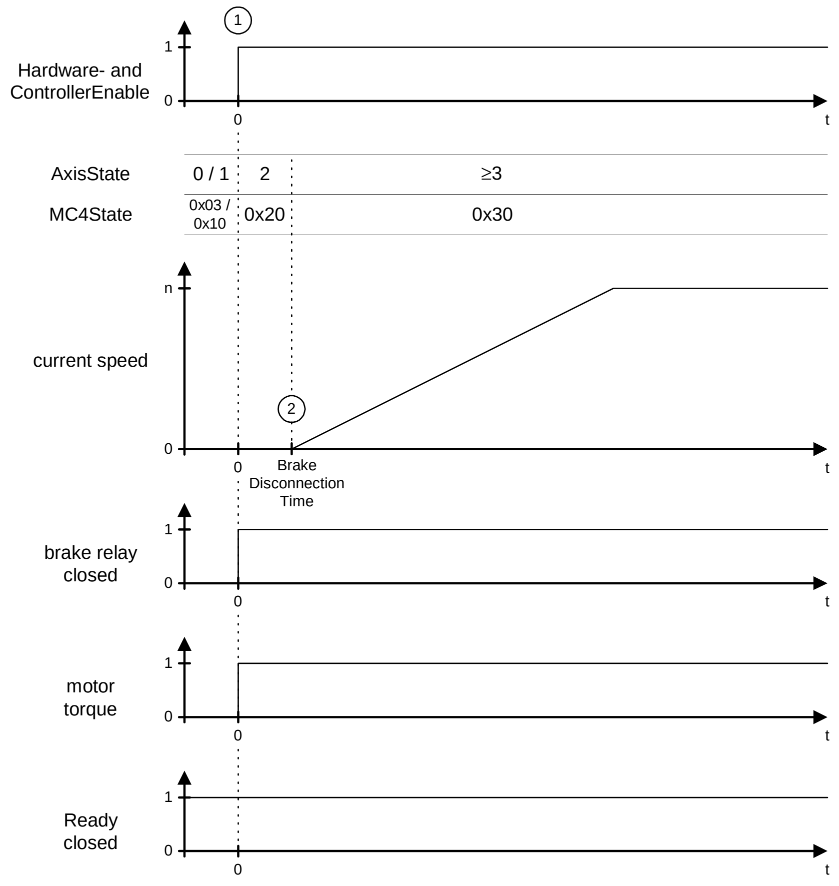


Fig. 10-1: Time diagram: Switching on enable

Before switching on enable, *AxisState* and/or *MC4State* depend on the enable signal that is being awaited. If hardware enable is missing, *AxisState* stands at 0 and *MC4State* at 0x03. If the drive is awaiting enable with *ControllerEnable*, *AxisState* is at 1 and *MC4State* at 0x10.
When switching on hardware enable and *ControllerEnable* (time 1), the brake relay is closed (brake lifted) and the motor torque is switched on, if there is no error message, the DC-circuit is loaded and all phases are connected. After the *BrakeDisconnectionTime* (time 2; default value: 100 ms), positioning jobs can be sent to the drive.

10.2 Lifting Enable

When lifting enable, three events may occur.

10.2.1 Ramping-down within Maximum Ramping-down Time

When lifting enable, the following behavior over the time occurs if the drive is ramped down within the maximum ramping-down time:

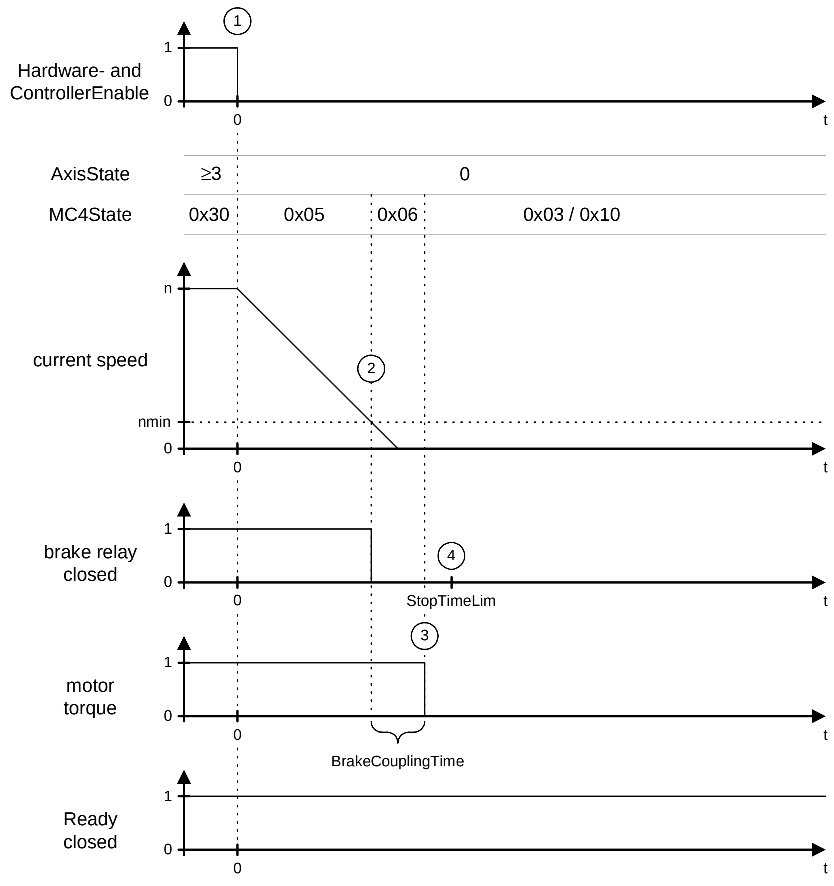


Fig. 10-2: Time diagram: Ramping down within maximum ramping-down time

When lifting enable (time 1), the drive ramps down with maximum current. The drive comes to a stop before expiry of the maximum ramping-down time *StopTimeLim* (time 4). As soon as the acutal velocity drops below the velocity limit (time 2: actual velocity < nmin (10 min^{-1})), the brake relay drops. After expiry of the BrakeCouplingTime (time 3), the motor is switched torque-free. The ready contact remains closed.
Thus error-free ramping-down is executed if *CoastOption* is not active for lifting enable (*bEnableCoast* = FALSE).

10.2.2 Stop Time Limit Exceeded, Drive Is Brought to a Controlled Stop

When lifting enable, the following behavior over the time occurs if the drive is not ramped down within the maximum ramping-down time, but the drive can still be brought to a controlled stop:

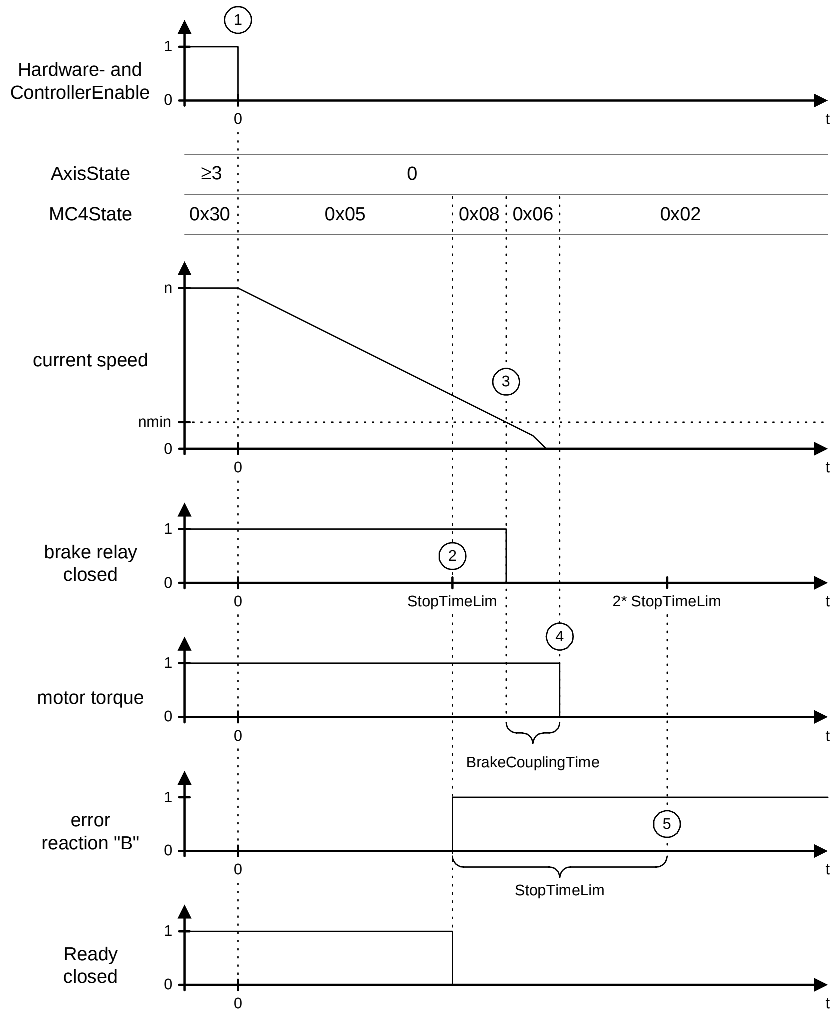


Fig. 10-3: Time diagram: Ramping down (stop time limit exceeded, bringing to a stop)

When lifting enable (time 1), the drive ramps down with maximum current. However, the drive does not come to a stop before the maximum ramping-down time *StopTimeLim* (time 2). For this reason, the error message “Stop time limit exceeded“ is triggered with error reaction "B". In case of error reaction "B", the ready contact is opened and the drive is still brought to a controlled stop with maximum current. If the drive comes to a stop (time 3) within the stop time limit of error reaction "B" (time 5), the brake relay is opened

(brake drops) and the motor is switched torque-free after the *Brake-CouplingTime* (time 4).
Thus, the drive is ramped down in a controlled way as long as the drive comes to a stop within the double stop time limit plus the brake coupling time and *CoastOption* is activated neither for lifting enable nor for error reaction "B" (*bEnableCoast* = FALSE; *bError-Coast* = FALSE).

10.2.3 Stop Time Limit Exceeded, Drive Brought to an Uncontrolled Stop

When lifting enable, the following behavior over the time occurs if the drive is not ramped down within the stop time limit and the drive cannot be brought to a controlled stop:

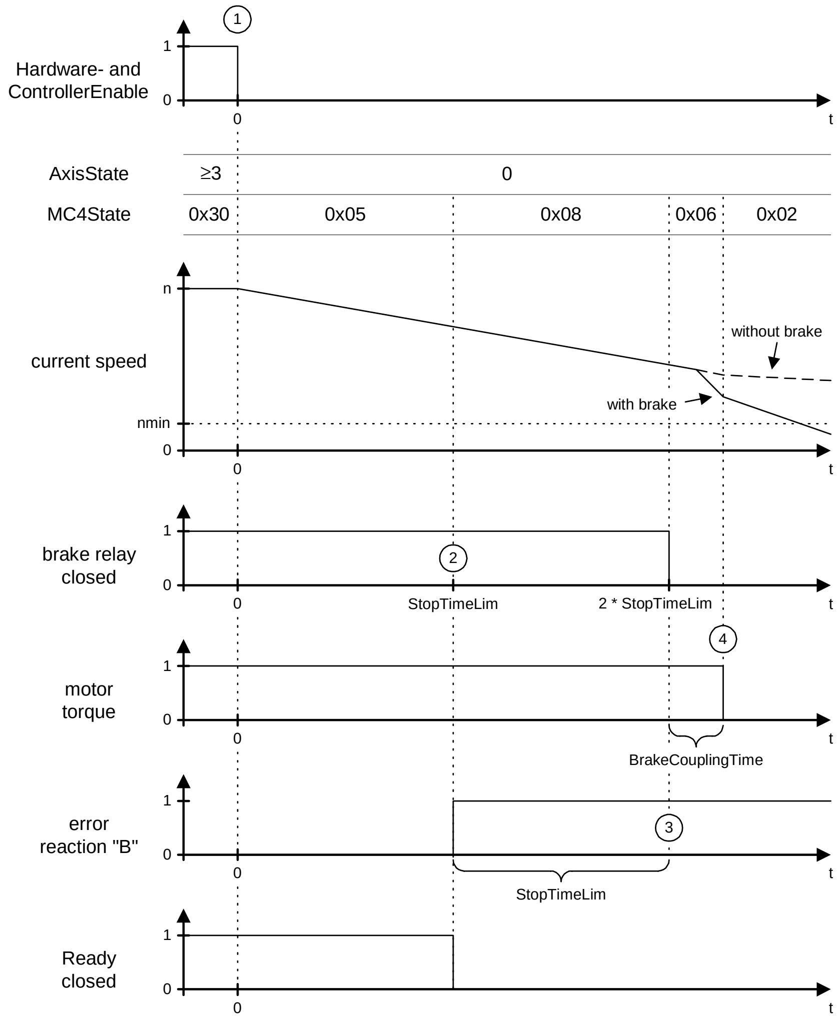


Fig. 10-4: Time diagram: Ramping down (stop time exceeded, drive coasts to a stop)

When lifting enable (time 1), the drive ramps down with maximum current. However, the drive does not come to a stop within the *StopTimeLim* (time 2). For this reason, the error message “Stop time limit exceeded“ is triggered with error reaction "B". In case of error reaction "B", the ready contact is opened and the drive is still brought to a controlled stop with maximum current. If the drive does not come to a stop within the stop time limit of error reaction “B“(time 3), the brake relay is opened (brake drops) and the motor is switched torque-free after the *BrakeCouplingTime* (time 4). Thus, the drive is ramped down in a controlled way as long as the drive comes to a stop within the double stop time limit plus the brake coupling time. This applies only if *CoastOption* is neither active for lifting enable nor for error reaction "B" (*bEnableCoast* = FALSE; *bErrorCoast* = FALSE). The uncontrolled stop stop will not trigger any error message.
The behavior of the drive in case of an uncontrolled stop depends on whether the motor has a holding brake or not.

10.3 Error Reaction "B"

In case of an error reaction "B", two procedures may occur.

10.3.1 Ramping down within Stop Time Limit

In case of an error with error reaction "B", the following behavior occurs if ramping-down is completed within the stop time limit:

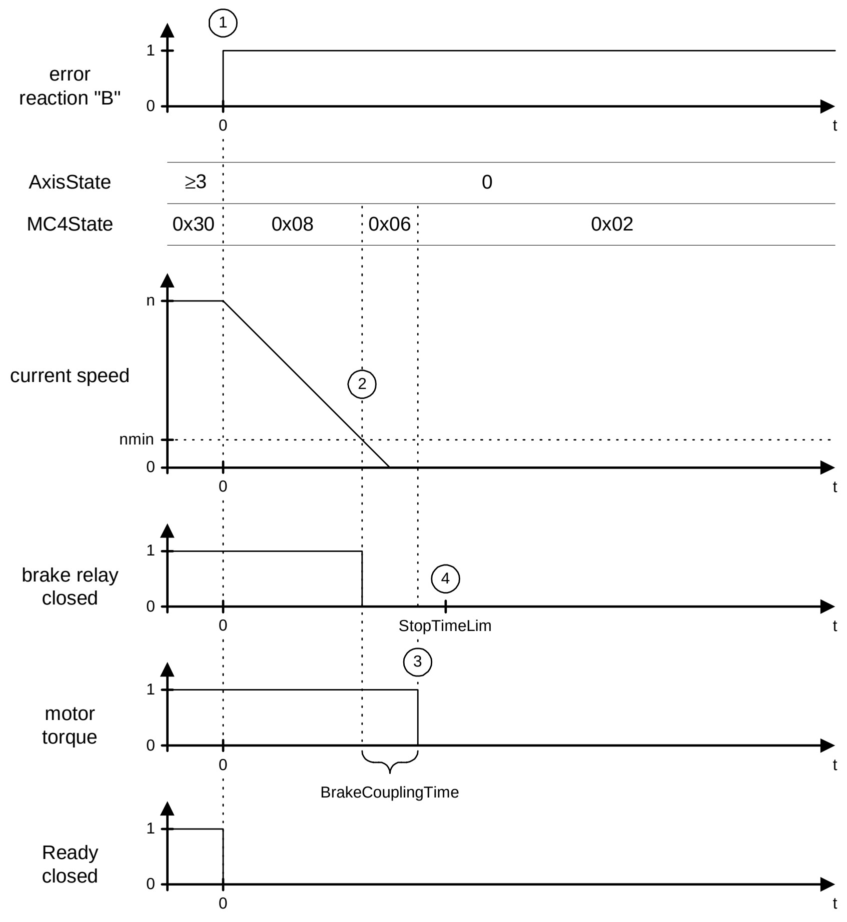


Fig. 10-5: Time diagram: Error reaction B / Ramping down within stop time limit

In case of an error with error reaction "B" (time 1), the drive ramps down with maximum current and the ready contact is opened. The drive comes to a stop before expiry of the StopTimeLim (time 4). As soon as the actual velocity drops below the velocity limit (time 2: actual velocity < nmin), the brake relay drops. After expiry of the *BrakeCouplingTime* (time 3), the motor is switched torque-free. Thus, the drive is ramped down error-free if *CoastOption*is not active for error reaction "B" (*bErrorCoast* = FALSE).

10.3.2 Stop Time Limit Exceeded

In case of an error with error reaction "B", the following behavior occurs if ramping-down is not completed within the stop time limit:

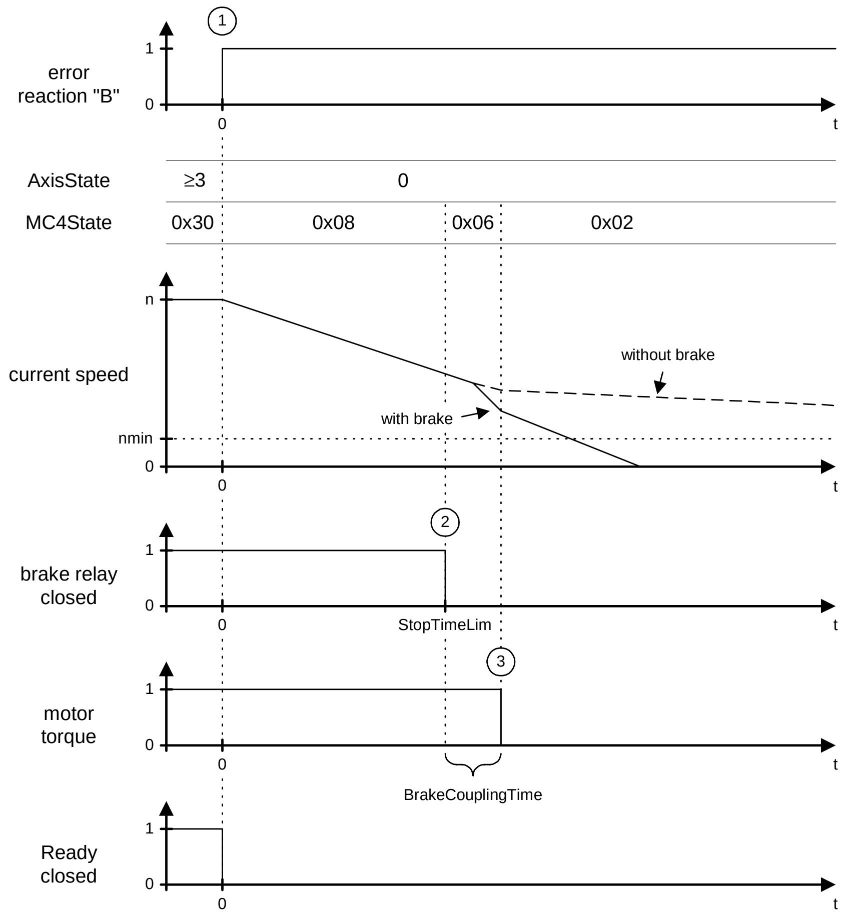


Fig. 10-6: Time diagram: Error rection B / Stop time limit exceeded

In case of an error with error reaction "B" (time 1), the drive ramps down with maximum current and the ready contact is opened. However, the drive does not come to a stop before expiry of the *StopTimeLim* (time 2) (actual velocity < nmin). Therefore the brake relay is opened and the error message “Stop time limit exceeded“ is triggered. As an error with the same error reaction is already active, this error message has no effect on the drive. After the BrakeCou-plingTime (time 3), the motor is switched torque-free.
Thus, the drive is ramped down in a controlled manner for the stop time limit period plus the brake dropping time and then brought to an uncontrlled stop (drive coasts to a stop). This only applies if *CoastOption* is not active for error reaction "B" (*bErrorCoast* = FALSE). The uncontrolled stop generates no error message.
The behavior of the drive in the event of an uncontrolled stop depends on whether the motor has a holding brake or not.

10.4 Error Reaction "A"

An error with error reaction "A" results in the following behavior:

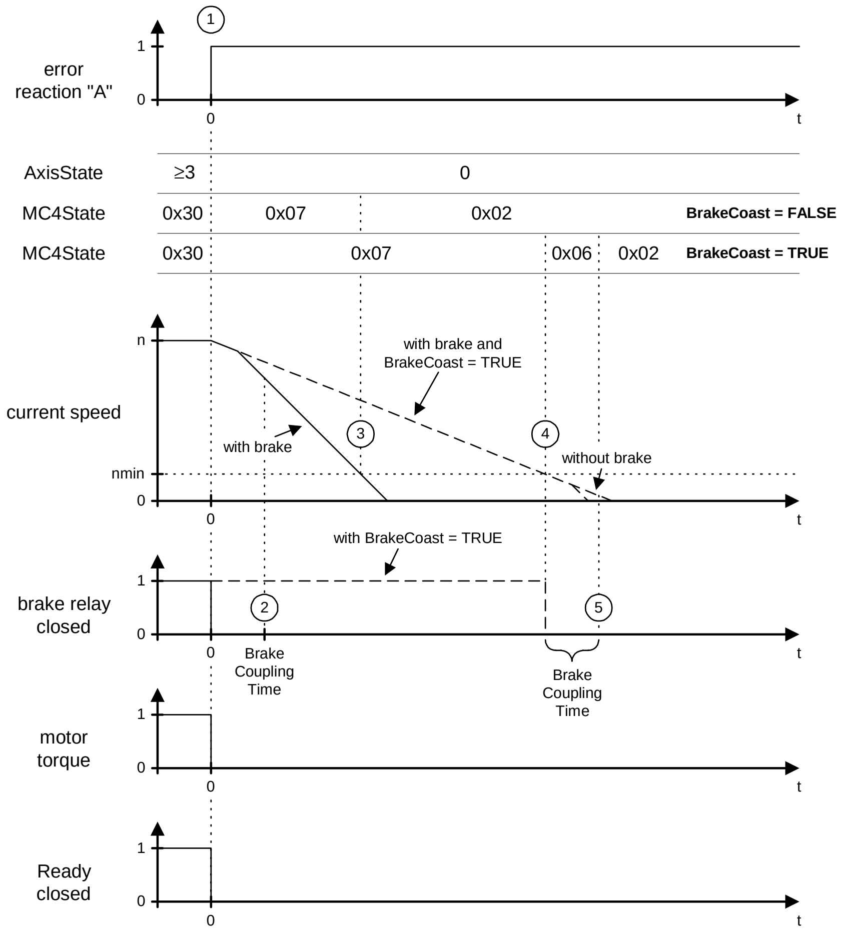


Fig. 10-7: Time diagram: Error reaction A

In case of an error with error reaction "A" (time 1), the drive is switched torque-free immediately, the brake drops and the ready contact is opened. When the *BrakeCouplingTime* is over and the drive comes to a stop (actual velocity < nmin) (time 3), the MC4State is switched to 0x02. From that state, the drive can be controlled again after quitting the error message.
With the *CoastOption* for the brake, the brake will not drop until the drive is standing (time 4: actual velocity < nmin). After the Brake-CouplingTime the MC4State is switched to 0x02.
The same reaction is activated if the CoastOption for enable is activated when lifting enable (*bEnableCoast* = TRUE) or the *CoastOption* for error reaction “B“ is activated for an error with error reaction "B" (*bErrorCoast* = TRUE).

PD_UserMan_ZeitDiag_us.fm

The behavior of the drive depends on whether the motor has a holding brake or not.

10.5 Overload Switchoff

In principle, two procedures may occur in case the system switches off due to overload.

10.5.1 Overload Switchoff within Maximum Switchoff Time

In case the system switches off due to overload, the following behavior over the time occurs if the overload position is reached within the maximum stopping time:

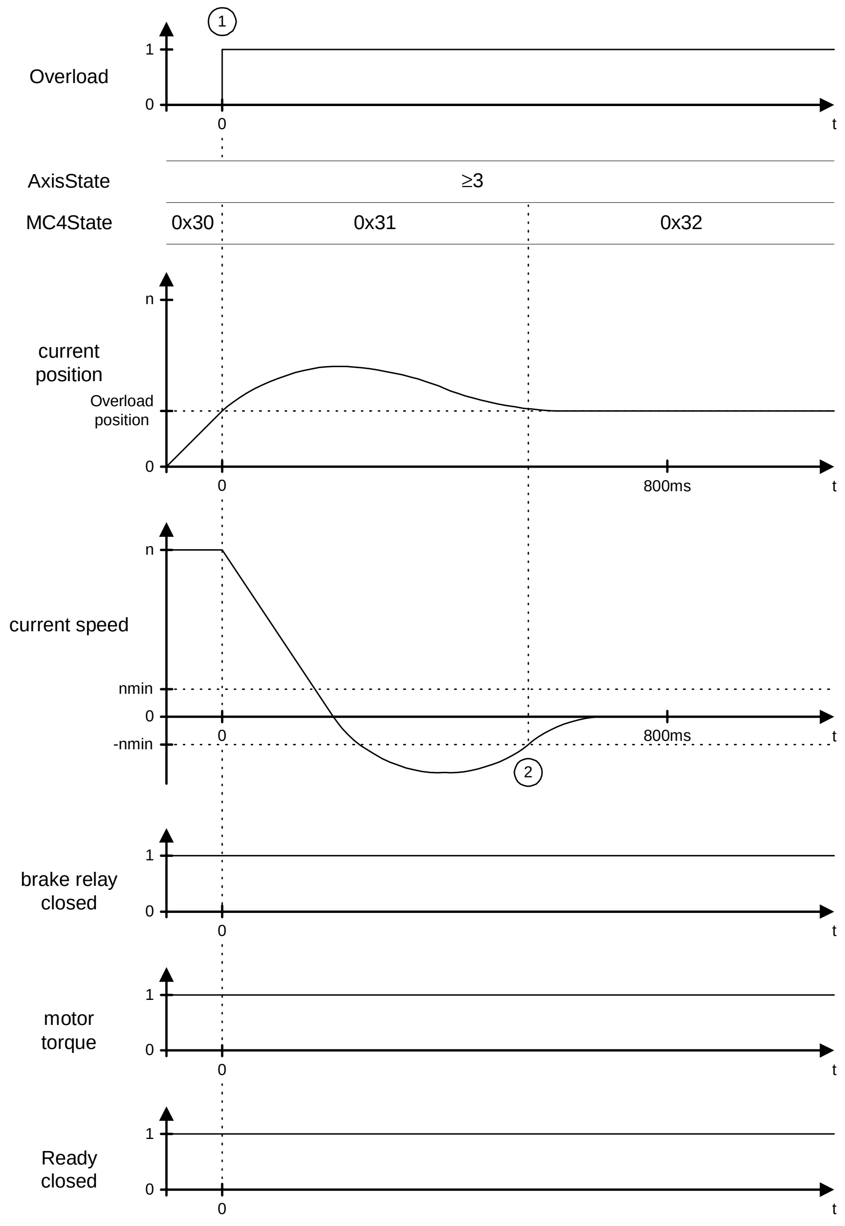


Fig. 10-8: Time diagram: Overload within maximum stopping time

When the system switches off due to overload (time 1), the drive is controlled with reference to the overload position. The overload position is the actual position at the time when the overload switch-off occurs. If the drive reaches the overload position within the maximum stopping time of 800 ms and the motor comes to a stop (time 2: |actual velocity| < nmin), the MC4State is switched to 0x32. After overload was quitted in this state, the drive follows the set values again.
This is a way to bring the motor to a stop without any error in case the system switches off due to overload.

10.5.2 Overload Switchoff with Transgression of Maximum Stopping Time

In case the system switches off due to overload, the following behavior over the time occurs if the overload position is not reached within the maximum stopping time:

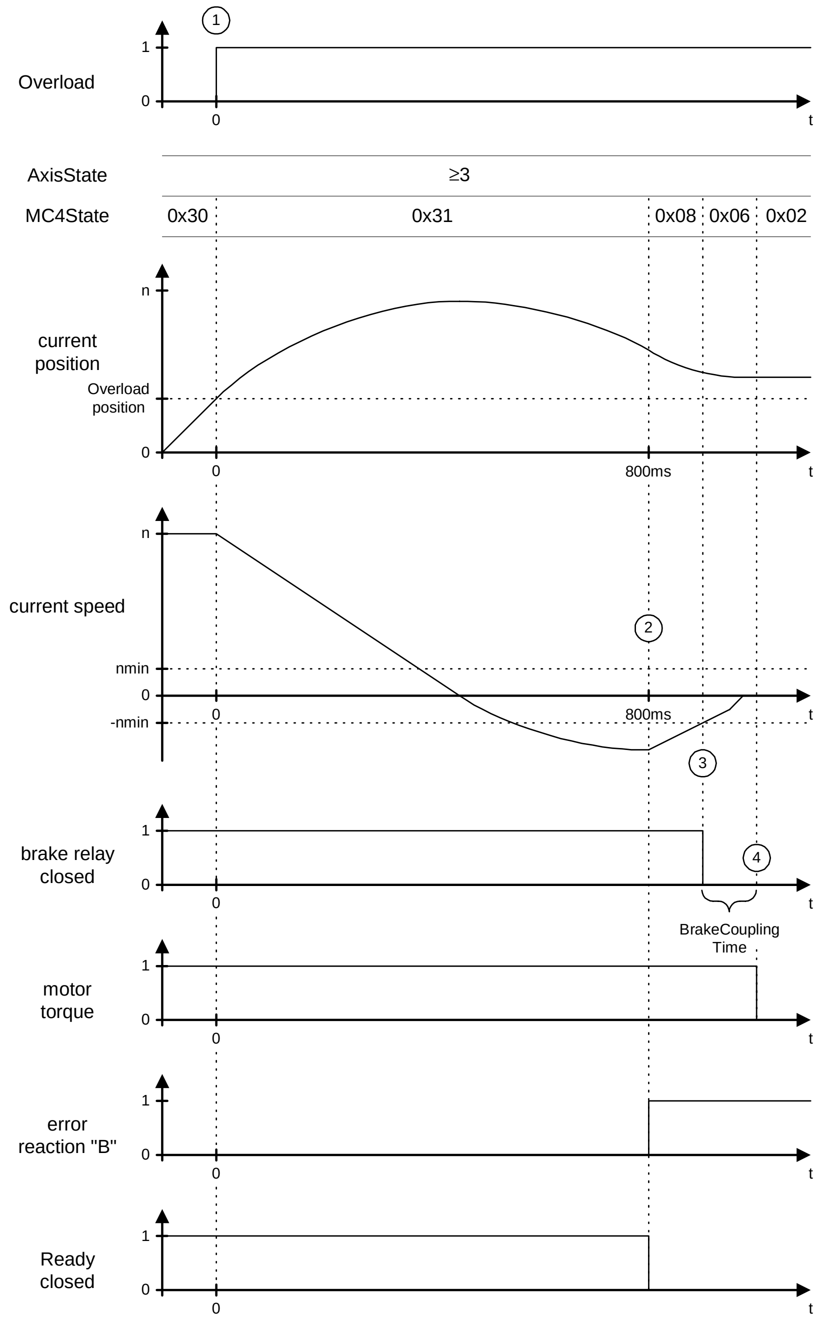


Fig. 10-9: Time diagram: Overload / Maximum stopping time exceeded

When the system switches off due to overload (time 1), the drive is controlled with reference to the overload position. The overload position is the actual position at the time when the overload switch-off occurs. If the drive does not reach the overload position within the maximum stopping time of 800 ms (time 2), the error message “ramping-down time exceeded“ is triggered with error reaction “B“ and the motor is brought to a controlled stop. As soon as the actual velocity drops below the velocity limit (time 3: |actual velocity| < nmin), the brake relay drops. After the *BrakeCouplingTime* (time 4) the drive is switched torque-free.
The drive is brought to a stop in this way if *CoastOption* for error reaction "B" is not active (*bErrorCoast* = FALSE).

LEGEND

Absolute values are noted directly on the axes. They give a value referring to the zero point of the axis.

Differences are noted in curly brackets. The bracket gives the corresponding range.

11 Version Numbers and Compatibility

11.1 Version Designation in General

VXX.YY.ZZ

XX	- incompatible changes
YY	- upwardly compatible changes
ZZ	- compatible changes

11.2 Compatibility EPAS-4 - MAx-4

- EPAS-4 XX must be equal to MAx-4 XX
- EPAS-4 YY must be greater than or equal to MAx-4 YY
- EPAS-4 ZZ has no significance

11.3 Compatibility MAx-4 - MC-4

- MAx-4 XX must be equal to MC-4 XX
- MAx-4 YY must be greater than or equal to MC-4 YY
- MAx-4 ZZ has no significance

11.4 Compatibility MAx-4 - IEC-Libraries

- MAx-4 XX must be equal to Lib XX
- MAx-4 YY must be greater than or equal to YY
- MAx-4 ZZ has no significance

12 ENI-4

This chapter will give you an overview of the function, structure and operation of the engineering interface (ENI).

Further documentation:

- ENI Server user manual
- Online help for ENI Server, ENI Admin, ENI Control (parts of the ENI Server Suite) and ENI Explorer
- EPAS-4 Automation Toolkit online help

12.1 Function of the ENI

The world of automation is so multifarious that it is impossible to enforce one standard for all software and hardware products. Users do not want to be committed to a small number of manufacturers, but demand the best solution to every problem.

Systems should become more transparent, they have to offer open interfaces to which users can connect their software and hardware without any problem. Within the PacDrive system, there are several interfaces offering that opportunity. Via OPC server, for example, you can directly access the data of the PacController. The most recent interface of the PacDrive system is the EPAS-4 Engineering Interface (ENI).

The ENI was developed as a client/server system. The server is an independent program based on a data storage system, preferably a version management (e.g. Microsoft Visual SourceSafe or MKS Source Integrity). All engineering data, such as program function blocks, variable lists, visualizations or configuration are stored via the server. The data storage format is XML. With this interface, other tools can now access all data of a project developed with EPAS-4. Those tools can read, generate and manipulate data.

Three examples are to help you better understand the ENI:

- Multi-user operation
- Version management
- Access by external clients

12.1.1 Multi-User Operation

The ENI permits multi-user operation in the implementation of PacDrive projects. In the past it was very difficult for several developers to work together on projects developed with a development environment for controllers. Although individual parts of a project could be exchanged by means of export and import functions, developers had to rely on verbal arrangements in order to coordinate their work.

The ENI offers a much more elegant solution to that problem. The project is stored entirely in the ENI. When a developer wants to work on a component, he loads it to his local working environment and edits it there. Other developers cannot edit that component at the same time. They can only access it after it has been released by the programmer. This strategy permits coordinated and thus effective work with several developers on the same project.

12.1.2 Version Management

The ENI supports the use of a source code version management program. Software is usually developed in several phases. The programmer will therefore implement the modules of a program step by step. Changes often have to be discarded because the way taken turned out not to be right after all. In the past, the developer saved his version and then made changes to a copy. If the changes were successful, he kept the copy; if they were not so successful, he returned to the original version. Everyone who has worked with this method knows that it needed improvement.

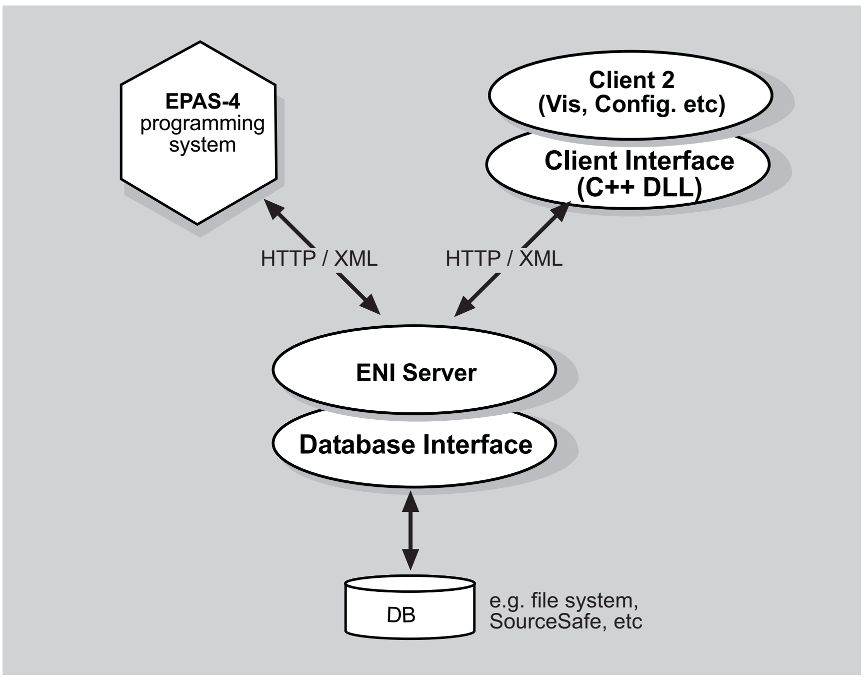
At least in the EPAS-4 development environment and modern development environments for many high languages, a solution has been found. Developers can now store different versions of their program or even individual components in the ENI database. The developer can retrieve those versions at any time, provided that the ENI uses a database that supports version management. The use of a version management does not interfere with the workflow, as it integrates smoothly with the EPAS-4 development environment.

12.1.3 Access by External Clients

In the past it was not possible to directly access the data of the development environment with other tools. With the ENI, any clients now have access to the data of the automation project. Those data can be read and even manipulated by the external client. For the programming language C++ there is a client interface (C++ DLL) to program a client for the ENI Server.

12.2 Structure and Communication of the ENI

The ENI operates a data storage system for PacDrive project data objects. This can be an existing database system or a local file system. The ENI interface is divided into a server and a client part, so that the storage system can also be located on another computer (otherwise multi-user operation would not be possible).



PD_EniKonzept_us.fh8

Fig. 12-1: ENI server concept

ENI Server

Engineering Interface Server runs as a separate service. The communication protocol is HTTP with XML as useful data.

PD_UserMan_ENI_us.fm

ENI Client

The clients of the ENI Server are e.g. EPAS-4, a CoDeSys programming system or other applications that need to access the data storage. All clients are equal.

Connection between ENI Server and database system (DB)

The ENI defines no own storage format. It is connected to an existing data handling system via drivers (database interface). Drivers are available for the database systems 'Visual SourceSafe 6.0', 'MKS Source Intergrity' and a local file system. For connection to other storage systems, please contact us.

Which data storage system an ENI Server is actually serving is defined in the ENI Server administration (See also „The ENI in EPAS-4“ on page 251.). The clients of the server can then access exactly that system.

HTTP as communication protocol, XML as data format

The ENI Server is accessed via the HTTP protocol; the transmitted data (objects) are shown in the universally accepted standard format XML. The HTTP protocol makes it possible to work across firewalls and execute certain operations by means of standard tools.

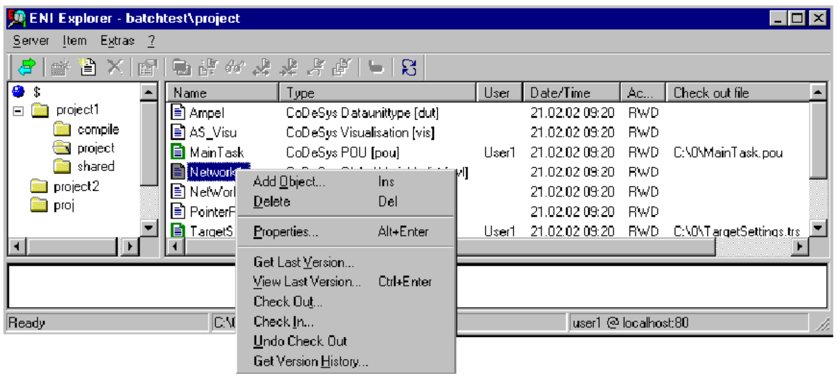
There is a client C++ DLL (client interface) to encapsulate the protocol in C++ classes, so that it is not a must to use only HTTP and XML software to access the ENI Server.

12.3 Structure in the Data Storage System / ENI Explorer

The ENI Server manages components created by EPAS-4 or other clients in a folder structure in the data storage system. There they are held as objects characterized by a type and access rights. Moreover, a version history is kept for each object. Within a folder, objects must be unambiguous by name and type.

ENI Explorer

ENI Explorer exists analogous to Windows Explorer as an independent program that can be connected to the desired ENI Server by means of suitable ENI access data. The folder/object structure in the ENI and the desired database functions can be called up right there, independently of the data handling system used and of EPAS-4. Object type, access rights, actual check-out status and users are shown as well.



Name	Type	User	Date/Time	Ac...	Check out file
Ampel	CoDeSys Dataunittype [dut]		21.02.02 09:20	RWD	
AS_Visu	CoDeSys Visualisation [vis]		21.02.02 09:20	RWD	
MainTask	CoDeSys POU [pou]	User1	21.02.02 09:20	RWD	C:\0\MainTask.pou
Network	vl]		21.02.02 09:20	RWD	
NetWorl			21.02.02 09:20	RWD	
PointerF			21.02.02 09:20	RWD	
TargetS		User1	21.02.02 09:20	RWD	C:\0\TargetSettings.trs

Fig. 12-2: ENI Explorer

12.4 The ENI in EPAS-4

The EPAS-4 programming system is a possible client of the ENI Server. Every part of a PacDrive project can be created as an object in the data handling system. For that purpose it is assigned to one of the four object categories that are possible in EPAS-4 (see below). The object storage is complete, i.e. the properties of the component, such as access rights, are transmitted in XML format along with the contents of the component. An automatic exchange with the data handling system can be defined for each object category.

XML format for EPAS-4 objects

The example below shows the depiction of a component in XML format. It includes e.g. the name of the component (<name>), its type (<pou>), its path in the EPAS-4 Object Organizer structure (<path>), access rights for the eight possible workgroups (<accesslevels>) and the contents of the component in the declaration part (<interface>) and the program part, which in this case is created in structured text (<st>,<body>).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<pou>
  <accesslevels> rw,rw,r,rw,r,rw,rw,rw </accesslevels>
  <path> \/languages </path>
  <name> ST_EXAMPLE </name>
  <flags> 4 </flags>
  <interface>
    <![CDATA[PROGRAM ST_EXAMPLE
    VAR
      YVAL: INT := -250;
      BOTTOM: INT := -250;
      RUN_STRING: STRING(20) := 'Start';
```

PD_UserMan_ENI_us.fm

```
        OFFSET: INT := 2;
      END_VAR
    ]]>
    </interface>
    <st>
      <body>
        <![CDATA[RUN_STRING:='Start';
        IF (YVAL < 0) THEN
        YVAL := YVAL + OFFSET;
      BOTTOM := YVAL + OFFSET;
        END_IF;
      ]]>
    </body>
    </st>
  </pou>
```

Categories for EPAS-4 ENI objects

There are four kinds of data objects for management via the ENI interface:

1. Project-specific objects: Components created specifically for a certain project.
2. Common objects: Components that are commonly used in several projects, e.g. function block libraries.
3. Automatically generated compilation objects: This has currently no significance and is not supported.
4. Local objects (not managed in the data handling system)

An object is assigned to one of the categories in the EPAS-4 project options and object properties.

The ENI parameters can be configured separately for each of the categories 1-3 (see below, Operation...).

Structure of the PacDrive project in the data storage system

One or several paths are created for a project in the connected data storage system and thus "in the ENI" under the "root" configured in the ENI control - folders where e.g. project components are shown as objects (for example, you could define a separate folder for each ENI object category).

Every project object receives a data type. Every component name taken over from EPAS-4 receives a type-specific extension and a type symbol and thus represents the object outside the programming system.

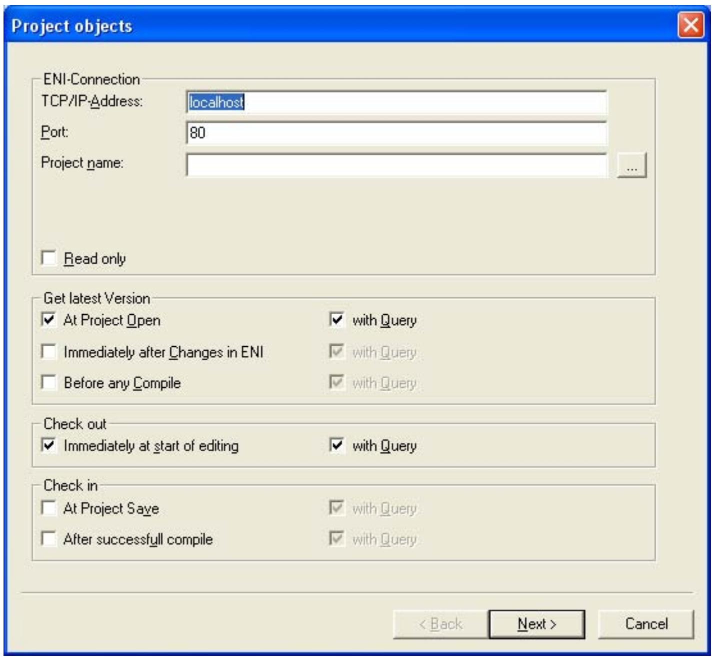
Operation of the ENI interface in the programming system

In EPAS-4, the ENI can be activated as a project option. Then the following functions are available for dialogs:

1. Definition in the project options if and to which ENI object category a newly created object is to be assigned.

2. Configuration of the ENI connection type in the project options for each of the three ENI object categories (project objects, common objects, compilation files; see above):

- Access data: port, access rights, project in the ENI storage to which the actual project is connected
- Definition which exchange with the data storage system is to take place automatically, and when (e.g. always call up objects from the database when a project is opened or check out whenever a change is begun in EPAS-4, etc.)


Project objects

ENI-Connection
TCP/IP-Address: localhost
Port: 80
Project name:
☐ Read only

Get latest Version
☑ At Project Open — ☑ with Query
☐ Immediately after Changes in ENI — ☑ with Query
☐ Before any Compile — ☑ with Query

Check out
☑ Immediately at start of editing — ☑ with Query

Check in
☐ At Project Save — ☑ with Query
☐ After successfull compile — ☑ with Query

Fig. 12-3: ENI settings

3. Individual assignment of an object to an object category (see above) in the object properties of each object

4. Entry of username and password via a database login dialog (project menu) for access to the data handling system

5. Database commands in the project menu, for individual objects or for the entire project:

- Call / call all

- Check out / check out all

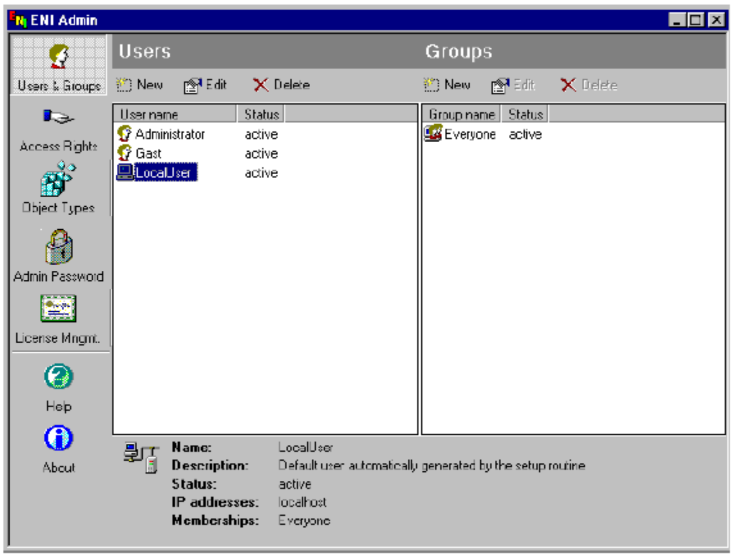
- Check in / check in all

- Undo check out / ... for all objects
- Version history for object / project
- Show changes
- Label version
- Insert common objects (from the database system into the local project
- Update status

12.5 ENI Admin and ENI Control

Certain settings can only be made locally on the ENI Server. The programs 'ENI Admin' and 'ENI Control' are available for that purpose. Like ENI Server and ENI Explorer, they are installed with the ENI Server Suite, which is shown with an icon on the system bar of your computer.

ENI Admin (ENIAdmin.exe) requires the entry of the administrator password to start. Users, user groups, access rights and the administrator password are defined here, licenses are managed and object data types shown.



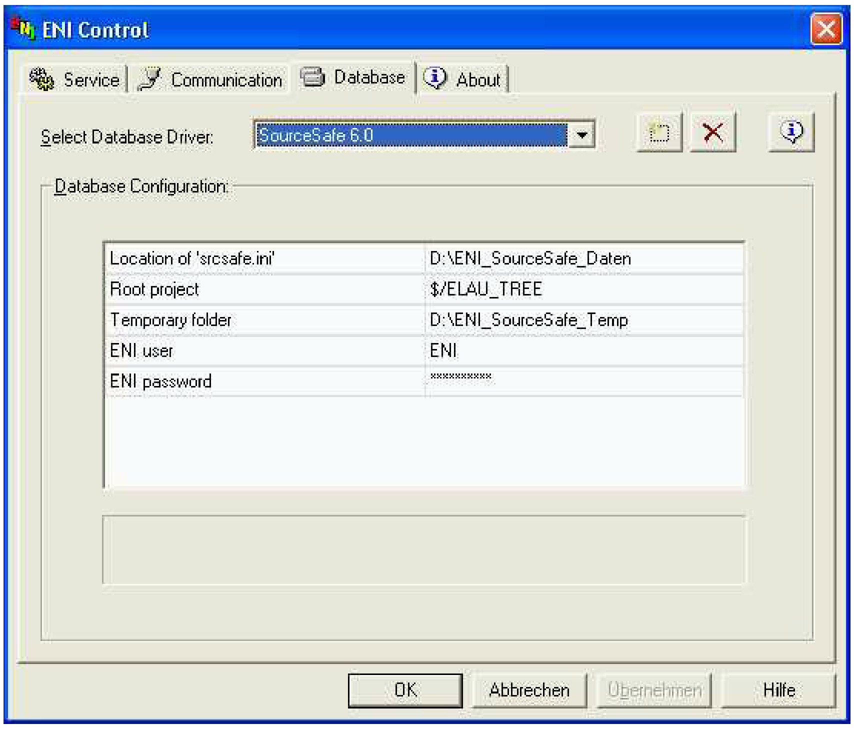
User name	Status
Administrator	active
Gast	active
LocalUser	active

Group name	Status
Everyone	active

Name:	LocalUser
Description:	Default user automatically generated by the setup routine
Status:	active
IP addresses:	localhost
Memberships:	Everyone

Fig. 12-4: ENI Admin

ENI Control (ENIService.exe) also requires the ENI administrator password to start. It is used to set the desired database driver even during the installation via setup. Moreover, it offers the opportunity to deliberately stop and restart the ENI service, which is usually started automatically, or to let the service run via another user account. In addition, certain communication parameters (communication timeout, port, etc.) of the server-client connection can be changed. An event log is available as well.



Select Database Driver: SourceSafe 6.0

Database Configuration:

Location of 'srcsafe.ini'	D:\ENI_SourceSafe_Daten
Root project	$/ELAU_TREE
Temporary folder	D:\ENI_SourceSafe_Temp
ENI user	ENI
ENI password	**********

Fig. 12-5: ENI Control

13 Frequently Asked Questions

What are VAR_IN_OUT variables?

VAR_IN_OUT variables are transmitted as pointers. Thus, the data need not be copied into and out of an instance. Located boolean variables cannot be transmitted as VAR_IN_OUT.

Are there pointers?

IEC 61131 knows no pointers. However, pointers are supported in the PacDrive M system. If you use pointers, we cannot assume any warranty that the programs work correctly - the user is entirely responsible himself.

Are there structuring possibilities for the validity range of variables?

You can create local and global variables.

What language can be used for what purpose?

There is no clear method. Here is just a recommendation:

- SFC for step-by-step procedures
- ST for algorithmic calculations
- FBD for program calls
- LD e.g. for locks
- IL for bit processing

Which programming language is the fastest one?

The execution speed of the programming languages differs only marginally.

Is there an oscilloscope function?

Yes, there is trace recording.

How can I display an instance of an FB in online operation ?

The instance of an FB can be shown under PROJECTS | OPEN INSTANCE. To do this, the cursor in the Object Organizer (left window) must stand on the FB or the FB must already be opened.

Are there partial compilations or do I always have to compile the entire program?

PROJECT | COMPILE ALL always compiles the entire project. An online exchange only generates and loads the changed code. After a rebuild, online exchange is no longer possible. You have to load the entire program.

What happens in case of online change?

This method permits changes while the PLC is running. In case of online change, the delta code, i.e. the latest modifications, is created. Directly after login this code is loaded into the the control (in the runtime system) and stored there. When the changes are activated, the code is added at the end of a cycle. As the code now exists twice, the memory will be reorgnized during the following cycles.

What is OPC?

OPC stands for OLE for Process Control and represents a standard interface for the connection of various hardware components to HMI/SCADA applications.
For further details, see Operating Manual OPC Server (article number 17130073-001)

14 Glossary

Absolute position encoder

Every point of a journey is marked by an unambiguous signal. For systems with absolute position encoders, no homing drive is required after switching on.
Normally the encoder information is recorded with an additional code disk which records a maximum of 4096 encoder revolutions. An easier variant is to store the infomration with a built-in battery.

Acceleration time

Acceleration time of the servomotor/amplifyer combination without external inertia from standstill to rated speed at peak torque.

ActiveX

The designation *ActiveX* is a general term. It comprises the technologies known as OLE Control (*OCX*). However, the terms were rearranged in the scope of the definition of *ActiveX*. *ActiveX* is a technology developed by Microsoft in order to support interactive contents for the World Wide Web. To be regarded as *ActiveX* control, a *COM object* must offer certain *interfaces*.

Analog signal

Many process data are available as analog values. We differentiate bewtween a voltage or current signal. To enable a computer to read in an analog value, it requires an analog digital converter. The resolution of the signals depends on the accuracy of hte AD converters.

Applet

An applest is a program used primarily on Internet sites. It is written in the Java programming language and can therefore run on all platforms.

AS interface

AS interface (ASI) is designed for very easy networking in automation technology. ASI was optimized for the lowest field bus level and offers the opportunity to connect not only binary, but also intelligent slaves and simple, analog units. With regard to the data volume, ASI has a lower capacity than field buses and is no competition for them.

ASCII

ASCII is the abbreviation for American Standard Code for Information Interchange. It is represented in 7-bit mode.

Axis identifier

Logic address of an MC-4 MotorController (not RealTimeBusAdr!)

Baud

The transmission speed baud is defined as the number of bits transmitted per second.

Baud rate

The baud rate is the speed of data transmission and gives the number of bits transmitted per second (baud rate = bit rate).

Booting

The start procedure of a computer (e.g. PC) and operating system is also called booting.

Browser

A "Web browser" is a program that permits access to and display of Internet pages. The main purpose of web browsers is to download HTML documents and the corresponding pictures from the Internet and display them.

Bus

Common transmission channel to which all participants are connected; it has two defined ends.

Bus connector

Physical connection between station and bus line.

Bus system

All stations that are physically connected via a bus cable form a bus system.

Cam laws

See set value generation

CAN

CAN stands for Controller Area Network and was originally developed by Bosch and Intel as a bus system for vehicles. In the meantime, CAN has also proven suitable as a field bus in automation technology. Like practially all field buses, CAN is based on the OSI 7 layer model. CAN is normed in ISO 11898 only for OSI layers 1 and 2 and is based on a bus structure. The transmission speed can be selected in a range of 20 kBit/s to 1 Mbit/s. The application layer is defined by the layer-7 protocols based on CAN, e.g. DeviceNet over CANopen. Due to the access method CSMA/CA and object-oriented addressing, CAN is particularly suitable for setting up very efficient event-controlled systems. In the multimaster system CAN, data can be exchanged directly between any number of clients.

CE

The CE sign documents that the product meets all requirements of the applicable European directives.

Client

An application that turns to a server to have certain tasks carried out.

CLSID (with OPC)

The class ID is a worldwide unambiguous code for a certain *COM object*. Via the *CLSID,* a client can address a certain *COM object*. A *CLSID* consists of 128 bit. To receive an unambiguous *CLSID*, the time of generation and hardware information of the computer used for generation are included in the *CLSID*.

CNC

An abbreviation for computerized numerical control. CNCs are used primarily in machine tools.

Cold restart

Program start at which all variable ranges are (re-)initialized. In case of certain events, such a cold start can be initiated automatically or manually by the user.

COM

The **C**omponent **O**bject **M**odel is used by Microsoft to define a mechanism for the cooperation of software components within the Microsoft Windows operating systems. By means of *COM*, an application can use the services of other applications or components. *COM* is the technological basis for *ActiveX*, *OLE* and *DCOM*. *COM* describes accurately to the bit how components work together and thus, in principle, is not tied to a programming language or a certain operating system. By now, the COM standard is also available on other operating systems.

COM objects

COM objects are software components that comply with the *COM* component model and make their functionality available to a *client* via COM interfaces. COM objects are defined unambiguously by their *CLSID*.

Commentary

Text enclosed between brackets and asterisks (cannot be nestet!) to explain a program; not interpreted by the programming system.

Communication

Here: data/information exchange between configurations.

Consistent data

Data that belongs together in terms of content and must not be separated are called consistent data.

Control

A system in which signals are given only in forward direction from the input to the output. It is not checked and corrected if the actual values of the system match the set values.

ControlNet

ControlNet is an open control network fulfilling the requirements of real-time applications with high data throughput. It is used to connect controls, I/O modules, PCs, visualization systems (HMI) and other intelligent devices. ControlNet transmits data with 5 MBit/s and enables a deterministic, reproducible transmission of time-critical control data. Network access is controlled by the time slice algorithm "Concurrent Time Domain Multiple Access" (CTDMA).

Continuous Function Chart (CFC)

Continuous Function Chart (CFC) is a programming language which, similar to SFC, has Boolean and arithmetic elements that work simultaneously. The continuous function chart uses no net-works. The elements can be placed freely. The elements in the processing list include function block, input, output, jump, label, return and comment. The inputs and outputs of those elements can be linked by dragging the mouse. The connecting line is drawn automatically.

An advantage of the continuous function chart CFC is that feed-backs can be inserted directly.

CPU

Central Processing Unit, micro processor

CUL

Products bearing the CUL sign meet the safety standards for the Canadian market. Canadian safety standards may differ in detail from the US standards of UL.

Current regulator

The current regulator regulates the difference between set current value and actual current value to 0.

Cv

Designator for the rated speed -> see VDI 2143

Meaning:

Cv lower -> required drive torque due to static strain (spring force, gravity, usable force) lower

Cycle

One round of the application program (which is checked periodically).

Cycle time

The time an application program needs for a *cycle.*

DC-circuit

Rectified and smoothed power supply.

DCOM

The **D**istributed **C**omponents **O**bject **M**odel expands the COM standard, so that *COM objects* can be used across computer limits.
DCOM is an optimized protocol which enables a client to use a *COM object* on another computer. For this purpose, the calls to the *COM object* are “packed“ in data packages, transported via a network, unpacked again and finally the *COM object* is called. The use of remote objects is transparent to a *client*.

Declaration

Definition of variables and FB instances in a declaration block, giving the designator, the data type or FB type and, where applicable, the start values, value ranges and field properties.
The definition or programming of POUs is also called declaration because they are declared with their properties in the programming system.

Declaration block

Summary of *declarations* of a variable type at the beginning of a *POU*.

Derived data type

With the help of a type definition, you can create a user-specific data type the elements of which consist of its elements or of elementary and/or derived data types.

DeviceNet

DeviceNet is an open, manufacturer-independent field bus standard in automation technology for the connection of PLCs with “intelligent“ devices such as sensors, push buttons, I/O assembly units, simple operating interfaces and drives via one single cable.
DeviceNet is an application protocol (OSI layer 7) based on the Controller Area Network (CAN). It offers high reliability for demanding applications with a high number of I/O assembly units.
Depending on the kind and length of cable used, the transmission speed is 125 kBit/s to 500 kBit/s.

Dezentralized technology

In converter technology, one differentiates between centralized and decentralized technology. The difference lies in the power supply. In centralized technology, one supply module generates the power for the individual amplifyers. In decentralized technology, each amplifyer or MotorController has its own power supply.

Directly shown variable

Variable without further designator; corresponding to a *hierarchic address*.

Drive bus

Modern, leading-edge systems transmit the information between control and drives in digital form. The most important technical requirement for the drive bus is a strictly deterministic transmission of the positioning data.

Edge

A “rising“ edge means the 0 -> 1 transition of a logic variable. Accordingly, a “falling“ edge is the 1 -> 0 transition.

Electronic packaging machine

In an electronic packaging machine, a mechanical main shaft is replaced by an electronic main shaft with individual flexible electric motor drives.

Elementary data type

A standard *data typ*e defined by IEC 61131-3.

EMC

EMC (electromagnetic compatibility) is the ability of an electric or electronic device to function satisfactorily in its electromagnetic environment. Limits for interference emission and interference proofness are set in norms.

EN

Abbreviation for European norm. In the course of European harmo-nization, national norms (e.g. DIN or VDE) are replaced by European norms .

Erasing unit

Assembly units to reduce induced voltages, which occur when swit-ching off power circuits with inductivities.

Ethernet

Ethernet has been around since the early 1980s and is nowadays found in all network technologies. An end of the dominance of the Ethernet is not in sight. It is available for various cable media, fiber-optic cables and radio and offers data rates of 10 Mbit/s to 1 Gbit/s. All kinds and generations of components cooperate in a network with no problem. The principal disadvantage of the restricted deterministic transmission behavior is usually meaningless for most applications (e.g. HMI) and can be neglected in view of the immense transmission capacities.

FBL

Function block language, essentially equivalent to the classical function block diagram.

Field

Sequence of elements of the same *data type*.

Field bus

Conventionally, sensors and actuators are connected with a control or evaluation unit via an analog or digital signal. This technology requires a multiwire cable for each connection between sensor or actuator and control. In addition, a facility for switching on and off (I/O) must be provided in the control for each sensor or actuator. If a field bus system is used, all devices are connected to one bus cable. Instead of facilities for switching on and off, an interface module is used. The resulting advantages are obvious. Commissioning and service become easier, the plant is more flexible. As a rule, this means cost advantages compared with conventional solutions. The different requirements for field bus systems, the variety of techical soltions and last, but not least, aspects of corporate policy, resulted in a large number of different field buses and prevented one single international standard. The devices of ELAU AG support the following international standards: CANopen, PROFIBUS DP, DeviceNet and ControlNet.

Following error

The following error is the dynamic distance between the computed set position value and the actual position value. The following error depends on the regulator structure and the amplifications of the individual regulator circuits. Minimal following errors can be achieved by means of feed forward for speed and current.

FTP

"FTP" (short for "file transfer protocol") is a transfer protocol used in the World Wide Web or in local networks using TCP/IP as network protocol. "Anonymous FTP" is offered by several thousand Internet computers as a way for everybody to access data, even if the respective computer has no user entry.

Function block diagram (FBD)

The function block diagram is a graphically oriented programming language. It works with a list of networks. Each network contains a structure representing a logic or arithmetic expression, the call of function blocks, a jump or a return instruction.

Function call hierarchy

This function displays the call stack if there is an actual diagnosis message from the message logger.

global

extensive; here: within the entire configuration

Ground conductor

One or several conducting parts that have very good ground contact.

Ground contact

Electrically conductive connection between a phase and PE.

Grounding

Grounding means connecting an electro-conductive part to the ground conductor via a grounding system.

GSM

GSM is a European mobile communication standard for cellphones in the 900 MHz range, defined by the CEPT (Conférence Européenne des Administrations des Postes et Telecommunications). Due to their digital data transmission technology, the GSM data services permit the remote diagnosis of packaging machines via cellphone.

Hierarchic address

Physical connection address of the I/O modules of a PLC system.

HMI

HMI (Human Machine Interface) are systems to operate plants, to monitor processes and to save data in industrial production processes. Smaller packaging machines use simpler operating units, but complex machines often use efficient visualization systems on the basis of industrial PCs. Such systems depict the operational procedures in a machine as flow diagrams and thus facilitate transparency for their monitoring. Important operating data are recorded and shown graphically. If something should not run smoothly, an alarm will be triggered immediately.

Holding brake

Brake in the motor; may be used only if the motor is standing.

Hot restart

Program start at the same place where a power failure occurred, while the PLC was in “RUN“. All buffered data ranges of the program are retained and the program can continue running as if there had been no interruption. In contrast to a *hot start at the beginning of a program*, the duration of the interruption must lie within a certain interval, depending on the process. For this purpose, the PLC system must have a separately secured real-time clock, so that the program can calculate the duration of the interruption.

Hot start at program start

Program start like *hot restart* with the only differenece that the unit starts from the beginning of a progrm if a maximum duration time has been exceeded. With the help of a state flag, the application program can recognize this situation in order to be able to prepare a special setting of its data.

HTML

HTML means HyperText Markup Language. It describes the logic components of a document. As a markup language, HTML contains commands for marking typical elements of a document, such as headers, text paragraphs, lists, tables or graphic references.

Id

see Identifier

Identifier (Id)

Identifier of an object

IEC 61131

IEC 61131-3 is an international standard for programming languages of programmable logic controls (PLCs). Instruction list (IL), function block diagram (FBD), ladder diagram (LD), structured text (ST) and SFC (sequential function chart) are different languages of this norm.

Instruction list

IL (instruction list) is a widely used programming languages for PLC systems, similar to Assembler.

Indirect FB call

Call of an FB instance, the name of which was transmitted as VAR_IN_OUT parameter in a *POU*.

Incremental encoder

Encoder that reports its position by two signals offset by 90°.

Instance

Structured data set of an FB by *declaration* of a function block, giving the type of FB.

Instancing

Instancing means the allocation of variables, including name and data type, during declaration by the user. By declaring different variables for the same type of FB, a data copy of the FB is formed n the control memory for each instance thus created.

Interface

interface

Interfaces (OPC)

COM interface, non-equivocal by an identification code - collection of connected functions. *COM object* can have one or several interfaces. Must not be changed after publication.

Internet

The Internet is a decentralized network, i.e. it does not depend on a single computer. The network consists of a number of sub-nets; the network protocol is TCP/IP.

The Internet consists primarily of the following services:

- e-mail, the most widely used service
- World Wide Web, the second-most widely used service
- Usenet, also called newsgroups
- FTP, a data transmission service

Inverter Enable

Inverter Enable is a switching unit in the MC-4 MotorController and enables the operating mode “safe stop“. In this operating mode, the unintentional start of a stopped drive due to an error is prevented.

IP 20

Protection means according to DIN 40050: Protected against finger touch and against the intrusion of solid foreign objects with a diameter of more than 12 mm.

IP 65

Protection means according to DIN 40050: Protected against touch, dust and the intrusion of spraying water from any directions.

IP 66

Protection means according to DIN 40050: Protected against touch, dust and against the harmful intrusion of powerful water jets.

IP 67

Protection means according to DIN 40050: Protected against touch and the intrusion of water with a certain pressure when submerging.

ISDN

ISDN (Integrated Services Digital Network) is a telecommunication standard that is widely used in Germany and Europe. In ISDN, all information is transmitted in digital form. This makes it possible to transmit the signals for language, text, pictures and data through the same telephone line. ISDN not only offers the possibility of using conventional telecommunication services. It also supports new types of communication, e.g. picture phones. The high transmission speed makes ISDN the ideal transmission medium for remote diagnosis of packaging machines.

Ladder diagram (LD)

LD (ladder diagram) is a programing language to describe networks with simultaneously working boolean electromechanical elements such as contacts and coils.

Limit switch

Limitation switch in the travel of a machine.

Logic address

Each object and each parameter in the control configuration has an unambiguous address by which it can be addressed.

Main shaft

See electronic packaging machine

Mains filter

Facility to educt distortions from the power lines to PE.

Mass

Mass is the total of all connected inactive parts of an operating means that cannot take on a dangerous contact voltage even in case of an error.

Master

Master devices determine the data traffic on the bus. If it has bus access rights, a master is allowed to send messages without external request. Masters are also known as active participants.

Master cam

Main or basic cam

Moment of inertia

Mass moment of inertia of a servomotor in the unit kgcm2.

MotorController

An electronic unit containing an amplification and control circuits for position, speed and current of a servomotor. The output size, the power connection of the servomotor, are modified in dependence on the set input value, the set positioning value.

Multi-element variable

Variable of the type *Field* or *Structure*, composed of several *data types*.

Network

Unit which is marked by the fact that the content of the actual result is not retained beyond the network boundaries.

OCX

The term **O**LE **C**ontrol E**x**tension marks *COMobjects* which can depict themselves via a user interface. Such COM-based control elements are additions to the known Windows control elements such as button, checkbox, etc. The term “OCX“ has been replaced by the more general defintion “ActiveX Control“ and is no longer used.

offline

without connection

OLE

Object **L**inking and **E**mbedding is a *COM-based* mechanism for integrating different document types within a document. OLE makes it possible e.g. to embed a spreadsheet in a text and use the functionality of that spreadsheet program within the text processing program.

OLE Automation

OLE Automation defines a COM-based mechanism that enables script languages (macro languages of applications, Java script, Visual Basic, etc.) the easy use of *COM objects*.

OPC

OPC is the abbreviation for "OLE for Process Control". The OPC Foundation has the aim of transforming the different interfaces used in process control and automation applications into a standardized form in order to enable a smooth collaboration between control applications, the fiels systems and components and the business and office applications and thus enable a simple, quick and inexpensive system integration. Based on Microsoft’s OLE and COM/DCOM technology, the OPC standard offers manufacturers of hardware and software a possibility to develop a standard driver for data exchange with other systems, e.g. HMI or SCADA.

OPC groups

An *OPC group* is a logic unit for structuring *OPC items* used by a *client*. Groups are created by a client and can contain one or several *OPC items*. Group calls can be applied to the *OPC items* of a group.

OPC items

The *OPC items* represent the process variables. *OPC items* are identified by an ItemID and have a value, a state information and a time stamp.

OPC server

An *OPC server* is a *COM object* offering the *interfaces* of the OPC specification of the OPC-Foundation.

Open Control

The Open Control Foundation wants to develop future-oriented concpets for the standardization of PC-based automation solutions. For an automation solution, usually different hardware and software components have to be integrated in an overall system, e.g. a field bus connection for decentralized I/Os, a control, a visualization system and various projecting tools for PLC, field bus and HMI. By using a common data pool, the Open Control Foundation tries to achieve a usefulness of the projecting data throughout the system, to avoid errors and in general enable cost reductions in projecting.

Operand

Language element with which an operation is effected, e.g. %IX1

Operator

Symbol that represents an action with which an operation is executed, e.g. AND

Opto-coupler

Optical connection between two electrically independent systems.

PacDrive

The PC-based automation system PacDrive is ideally suited for the complete automation of packaging machines. The IEC 61131 PLC and motion tasks (VarioCam) are realized with a central PC-based intelligence. There are interfaces to all commercially available HMIs. The I/O level of the PLC is realized by standard field buses. The highly dynamical servo drives are connected via SERCOS. The built-in Ethernet interface with TCP/IP enables remote diagnosis via modem or Internet down to the motor current.

Peak torque

The maximum torque which the combination of servomotor and amplifyer can emit to the drive shaft of the servomotor for a short time.

PLC

Programmable Logic Controller

PLC

Abbreviation for **P**rogrammable **L**ogic **C**ontrol.

Plug-In

Plug-Ins (PI) extend an Internet browser by additional features, which make it possible to display varous file formats which the browser does not support by default.

POU

Program organization unit; IEC 61131-3 component of the type Function, Function block or Program, from which PLC programs are created hierarchically.

Position resolution

The position resolution says how many increments are available per 360 degrees motor revolution. In principle, positioning accuracy and regulating quality are the better, the higher the position resolution.

In general, resolvers have a position resolution of 4,096 to 65,000 increments per motor revolution. High-resolution SinCos encoders, as used in the PacDrive™ system, have a resolution of 1,000,000 increments per motor revolution.

PROFIBUS

PROFIBUS is a manufacturer-independent, open field bus standard for applications in manufacturing, process and building automation. Manufacturer-independence and openness are guaranteed by the international norm EN 50 170. PROFIBUS enables the trouble-free communcation between devices of different manufacturers without any special interface adjustments. PROFIBUS is suitable both for fast, time-critical data transmission and for comprehensive and complex communication tasks. PROFIBUS consists of a family of 3 compatible variants: PROFIBUS-DP, PROFIBUS-FMS and PROFIBUS-PA.

The PROFIBUS connections of ELAU AG are based on PROFIBUS-DP. PROFIBUS-DP is specially designed for the communication between automation systems and decentralized peripheral units on field level and is characterized by high speed and low connection costs. PROFIBUS-DP is based on a bus structure and uses a transmission technology according to RS-485 or fiber-optic technology. The transmission speed can be selected in the range of 9.6 kBit/s to 12 Mbit/s.

ProgID

The Program ID is a plain-text substitute for a *CLSID*. *ProgIDs* have the format “library.class.version“ and, in contrast to *CLSID,* are non-equivocal only by convention. Example of a *ProgID*: „OPC.Producer.20“.

Protocol

The rules for data transmission between computer systems and other devices are called protocol. A protocol defines the packeting of data, i.e. how many data bytes are transmitted at once, the treatment of data that were transmitted faulty (kind of error protocol), the channels availble for transmission (full duplex or half duplex method), giving the beginning and end of a transmission and the kind of compression of the data packages.

The ISO (International Standards Organization) created the OSI model (Open Systems Interconnection) as a reference model for all data transmission between electronic systems. According to the OSI reference model, a transmission protocol has seven leves.

Position regulator

The position regulator regulates the difference between set position and actual position to 0. Output: set speed.

Rated speed

Usable speed at rated torque. Idle-running speed and mechanical limit speed of the servomotor are higher.

Rated torque

The rated torque is the permanent torque in Nm of a servomotor/amplifyer combination at rated motor speed. Due to the losses, which are related to the speed, it is less than the standstill torque at speed 0. With an ambient temperature and depending on the thermal motor time constant, an overtemperature of 40°C occurs on the motor casing.

Real-time operating system

In real-time operating systems, like in conventional operating systems, several parallel jobs, i.e. tasks, threads or processes, are processed simultaneously. To fulfill the real-time requirements in the control job, real-time bus systems use special methods of computing time allocation. The predictability (determinism) of the system behavior, i.e. the guarnteed processing of a job within a given time frame, is an essential criterion for real-time systems. Moreover, the administration of system resources, such as common or exclusive memory ranges, IO devices, ports and time services as well as the task communication via semaphore, mailbox and event-queues put particularly high demands on the real-time operating system.

Real-time process

The tasks of the real-time process include e.g. processing the master and logic encoders, processing the motion tasks including calculation of set values, processing the measuring inputs, recording trace values and of course exchanging real-time data via the real-time bus (SERCOS).

Real-time system

According to a DIN definition, a real-time system is a system that replies in a defined way to an external event under all circumstances, with the reply being generated within a certain time frame (scan time). If a time frame is exceeded, this is regarded as a failure of the whole system. If the process is less critical, it may be admissible to exceed that time frame. In this case, we speak of “soft real time“. A soft real time system is only acceptable if exceeding the time frame causes no risk to man or machine. As a rule, pakkaging machines are systems that represent “hard“ real-time requirements for the control used.

Reference potential

Potential from which the voltages of the circuits involved are regarded and/or measured.

Regulation

A set value of a proces size is compared with the actual value and amplified and regulated in such a way that the difference is zero, if possible. In contrast to a control, the influence of a distortion is regulated out in a regulation. In principle, there is control and disturbance variable behavior. In a servo drive, position, speed and current are regulated.

Reset

Procedure that triggers the boot of a computer (e.g. activating the reset button, switching on the control voltage).

Resource

As a rule, the ressource is equivalent to a CPU.

RS-232

The RS-232-C interface is a standardized serial interface which is largely identical with the V.24 interface (CCITT). It is designed for asynchronous data exchange between a computer system and a peripheral unit.

RS-422

In functional terms, the RS-422 interface is equivalent to the RS-232 interface, but it uses a differential signal transmission and thus makes it possible to bridge greater distances even at higher transmision speeds while being little prone to distortion.

RS-485

The RS-485 interface is a standardized serial interface. In functional and electrical terms, it is largely equivalent to the RS-422 interface and permits the connection of several terminals to a connection cable (bus).

Restart

Summary term for *hot restart* and *hot start at program start*.

SCADA

SCADA systems (Supervisory Control and Data Acquisition) contain components for the control, analysis, monitoring, storage and administartion of the information flow between the systems of the field level and the management level of a company. This ensures the connection between the decentralized I/O units and the machine controls on the one hand and the office computers of the mangement level on the other.

Scan rate

Time in millseconds which tells in which time the set values of an axis are calculcated and compared to the actual values and then the differences are processed in the controller.

Sequential function chart

SFC (sequential function chart) is a programming language to describe sequential and parallel control procedures with time and result control.

Server

A server is a service provider. In connection with *COM*, the server is a *COM object* whose functionality (services) can be used by a *client* via its *interfaces*.

Servomotor

A servomotor nowadays is a brushless, permanently excited synchronous motor. A servomotor has a position encoder and is characterized by its extremly low internal inertia and high power density.

Servo amplifyer

see MotorController

Set value generation

Nowadays the motion functions of the individual axes of an electronic packaging machine are realized with intelligent software in the PacController. All motion laws known in VDI 2143 and some self-defined motion laws can still be modified online.

Short-circuit

Is a conducting connection arising due to an error between two conductors under operating voltage if there is no useful resistance in the fault current circuit.

Single-element variable

Variable that is based on one single data type.

Slave cam

Overlaying cam. A slave cam can only be started if a master cam is active.

Speed regulator

The speed regulator regulates the difference between set speed and actual speed to 0. Output: set power value.

Spline interpolation

Connection of individual points in space with a 3rd-grade polynomial.

Step

State knod in an SFC program in which commands of the *action* belonging to a *step* are triggered.

Standard functions

Number of *functions* defined firmly by IEC 61131-3 to realize PLC-typcial functionality.

Standard function block

Number of *function blocks* firmly defined by IEC 61131-3 to realize PLC-typical functionality.

Structured text

Structured text is a programming language for the description of algorithms and execution control by means of a modern high language.

Symbolic variable

Variable with designator to which a *hierarchic address* is allocated.

Syntax

Structure and structural cooperation of the language elements of a programming language.

TCP/IP

"TCP/IP" (abbreviation for "transmission control protocol/internet protocol") usually describes a whole family of protocols to connect computers in different networks. Nowadays TCP/IP is used in many LANs and is the basis for the worldwide Internet.

The IP takes over the transport of data, while the TCP takes care of their delivery.

Telecommunication connection

Telecommuncation in this case means the transmission of data between electronic data processing units or systems over a longer distance e.g. with the help of electric cables, fiber-optic cables or radio waves.

Teleservice

Teleservice enables the remote connection of controls of packgaing machines via telecommunication networks (ISDN, GSM) for service, maintenance, commissioning or machine conversions. For the machine builder, teleservice can guarantee worldwide short service reaction times and thus a high availability of packaging machines.

Terminal resistor

is a resistor to adjust the performance on the bus cable; terminal resistors are necessary at the ends of cables and/or segments.

Thermo protective contact

Temperature-sensitive switch built into the motor winding.

Torque constant

Quotient of standtsill torque and standstill current in the unit Nm/A.

Transition

Transition from one SFC step to the next by evaluating the transition condition.

Translation of functions

A *function* can be applied to entries of different *data types*.

Type definition

Definition of a user-specific *data type* on the basis of existing *data types*.

UL

Underwriter Laboratories Inc. (UL) is an independent testing and certifying organization in the USA. Products bearing the UL sign comply with the safety standards of the UL.

Ungrounded structure

Structure without galvanic connection to the ground. In most cases an RC module is used to carry off parasitic currents.

Variable

Name of a data storage, which can take on values determined by the *data type* and entries in the variable *declartion*.

VarioCam®

The VarioCam® Online Toolbox provides you with extensive functions, including standard cams. Lengthy modificatin and restart is not required, as the cams are simply reshaped by parameterization.

VxWorks

ELAU AG's MAx-4 PacController is based on the real-time operating system VxWorks® by Wind River Systems. VxWorks® is the market leader in the field of real-time bus systems and meets highest requirements with regard to real-time behavior, robustness, runtime and memory efficiency.

Windows

A strong trend towards Microsoft operating systems can be observed in automation technology. This development confirms the direction towards open, scalable automation solutions on the basis of the fast-developing PC hardware. Software becomes a decisive factor for automation solutions. The driving force in this development is the 32-bit world of the Windows operating systems, on which many producers of industrial software are now betting. The products usually include standard interfaces for graphic imports, data exchange (DDE, ODBC) and for the integration of applications (OPC). By means of interfaces, the communciation between different software products can be effected easily.

However, these standard operating systems from the office domain cannot fulfill the strict real-time requirements for a driving system. Windows CE, Windows NT and Windows NT Embedded in their original version can only be used for automation projects with “soft“ real-time requirements. Therefore ELAU AG's MAx-4 PacController is based on the real-time operating system VxWorks®, which guarantees absolutely deterministic behavior.Thanks to the integrated OPC interface, the MAx-4 PacController can easily be integrated in Windows environment.

WWW

The "WWW" (abbreviation for "world wide web") is a multi-media hypertext information system in the Internet.

Zero pulse

The zero pulse is issued by incremental encoders once per revolution. It helps to zeroize the machine.

15 Appendix

15.1 Contact

For repair

Please send the components to be repaired or checked, along with the error report, to this address:

ELAU AG

Abt. Kundendienst	house address:
Postfach 1255	Dillberg 12
97821 Marktheidenfeld	97828 Marktheidenfeld

Phone:	+49 (0) 93 91 / 606-142
Fax:	+49 (0) 93 91 / 606-340

Service team

Should you need to talk to a member of our service team or require on-site service, please contact:

ELAU AG
Dillberg 12
D-97828 Marktheidenfeld

Phone:	+49 (0) 9391 / 606 - 0
Fax:	+49 (0) 9391 / 606 - 300
e-mail:	info@elau.de
Internet:	www.elau.de

ELAU, Inc.
4201 West Wrightwood Avenue
Chicago, Illinois 60639 - USA

Phone:	+1 773 342 8400
Fax:	+1 773 342 8404
e-mail:	sales@elau.com
Internet:	www.elau.com

ELAU SYSTEMS ITALIA S.r.l.
Via Tosarelli 300
I-40050 Villanova di Castenaso (BO)

Phone:	+39 051 / 7818 70
Fax:	+39 051 / 7818 69
e-mail:	info@elau.it
Internet:	www.elau.it

NOTE

Further contact addresses you can find on the ELAU homepage (www.elau.de).

PDM_AnhangStd_us.FM

15.2 Further Literature

ELAU can provide you with these manuals and instructions on the PacDrive™ System:[1]

Project Manual

Art.No. 17 13 00 58 - 00x (DE, EN, FR)

Programming Manual

Art.No. 17 13 00 61 - 00x (DE, EN)

Operating Manual MC-4 MotorController

Art.No. 17 13 00 62 - 00x (DE, EN, IT, FR)

Operating Manual CAN L2

Art.No. 17 13 00 66 - 00x (DE, EN)

Operating Manual PROFIBUS-DP

Art.No. 17 13 00 67 - 00x (DE, EN)

Operating Manual SM Motor

Art.No. 17 13 00 68 - 00x (DE, EN, IT, FR)

Operating Manual EPAS-4

Art.No. 17 13 00 70 - 00x (DE, EN)

Operating Manual MAx-4 PacController

Art.No. 17 13 00 71 - 00x (DE, EN, IT, FR)

Operating Manual OPC-Server

Art.No. 17 13 00 73 - 00x (DE, EN)

Operating Manual Device Net

Art.No. 17 13 00 76 - 00x (DE, EN)

Operating Manual HMI Libraries

Art.No. 17 13 00 77 - 00x (DE, EN)

Operating Manual INC-4 Incremental Encoder Module

Art.No. 17 13 00 78 - 00x (DE, EN)

Operating Manual CANopen

Art.No. 17 13 00 79 - 00x (DE, EN)

Operating Manual VarioCam® Editor ECAM-4

Art.No. 17 13 00 80 - 00x (DE, EN)

Operating Manual PacNet Module PN-4

Art.No. 17 13 00 81 - 00x (DE, EN)

Operating Manual SR Motor

Art.No. 17 13 00 82 - 00x (DE, EN)

PDM_AnhangStd_us.FM

1. Art.No. -000 (DE) german -001 (EN) english -002 (IT) italian -003 (FR) french

Operating Manual BusTerminal BT-4/DIO1
Art.No. 17 13 00 83 - 00x (DE, EN)

Operating Manual TTS
Art.No. 17 13 00 88 - 00x (DE)

User Manual Automatic Controller Optimization
Art.No. 17 13 00 89 - 00x (DE, EN)

Operating Manual PacDrive SCL
Art.No. 17 13 00 93 - 00x (DE, EN)

Operating Manual PacDrive PS-4 und PacDrive PD-8
Art.No. 17 13 00 94 - 00x (DE, EN)

Operating Manual Evaluation Kit
Art.No. 17 13 00 95 - 00x (DE)

Operating Manual PacDrive Controller P600
Art.No. 17 13 00 96 - 00x (DE)

15.3 Product Training

We offer practical workshops and seminars.

Our experienced seminar leaders will enable you to make optimum use of the vast possibilities of the PacDrive™ system.

NOTE

Please contact us for further information or to order our seminar program. See also our homepage (www.elau.de).

15.4 Modifications

07 / 1999

- Terms e. g. MAx-4, MC-4 and PacDrive™ M modified
- Sales Regions updated
- Manual revised

12 / 1999

- Separation of the programming manual in reference and user part
- Revised to 00.06.00

01 / 2001

- Chapter „The Basics of IEC-61131“ new
- Revision of Chapter „Programming Guidelines“
- Revision of Chapter „Teleservice“
- FAQs new
- Glossary widens

10 / 2003

- Chapter "Networks" new
- Chapter "Teleservice" modem recommendation extended
- Chapter "Teleservice" "Settings on Remote PC" extended for Windows 2000
- Chapter "Programming Guidelines" extended
- Chapters "Characteristics", "Time diagrams" and "Version desi-gnations" taken over from Reference Manual
- Chapter "ENI-4" new

NOTE

The latest documentation and modification service on this product are available on the ELAU Homepage (http://www.elau.de).

15.5 Index

PDM_UserMan_usSIX.fm

PDM_UserMan_usSIX.fm

15.6 Form for Error Report

This error report is absoluteley necessary in order to enable efficient processing.

Send the error report to your ELAU representative or to:

ELAU AG, Abt. Kundendienst
Dillberg 12, D-97828 Marktheidenfeld
Fax: +49 (0) 93 91 / 606 - 340

Sender:

Company:	City:	Date:
Department:	Name:	Phone:

Details on the defective product

Product name: ....

Article number: ....

Serial number: ....

Software version: ....

Hardware code: ....

Parameter enclosed: yes [ ] no [ ]

IEC program enclosed: yes [ ] no [ ]

Details of the machine on which the problem occurred:

Machine producer: ....

Type: ....

Hours of operation: ....

Machine number: ....

Date of commissioning: ....

Producer/Type of machine control:

....

ELAU_FoStoerung_us.FM

Description of the problem:

....

....

....

Additional information:

Problem state:	**Causes:**	**Concomitant phenomena:**
[ ] persistent	[ ] unknown	[ ] mechanical problems
[ ] when commissioning	[ ] wiring error	[ ] failure of mains supply (24V)
[ ] occurs sporadically	[ ] mechanical damage	[ ] failure of PMC-2
[ ] occurs after about.....hours	[ ] moisture inside the unit	[ ] motor failure
[ ] occurs in case of shocks	[ ] encoder defective	[ ] broken cable
[ ] depends on temperature		[ ] insufficient ventilation
[ ] foreign object inside unit		

Does the switching cabinet have an air conditioning system?Y/N [ ]

Have similar problems occurred before on the same axis?
How often: ....

Did the problems occur on certain days or times of day?

....

further information:

....

....

....

....

....

....

....

....

....

....

....

....

....

ELAU_FoStoerung_us.FM