



Kuhnke Electronics Instruction Manual PROFIBUS

E 365 GB

8 November 1995 / 67.338

This manual is primarily intended for the use of the designing engineer, the project planning engineer, and the developing engineer. It does not give any information about delivery possibilities. Data is only given to describe the product and must not be regarded as guaranteed properties in the legal sense. Any claims for damages against us – on whatever legal grounds – are excluded except in instances of deliberate intent or gross negligence on our part. We reserve the rights for errors, omissions or modifications. Reproduction even of extracts only with the editor's express and written prior consent.

Table of contents

1. Introduction	1-1
1.1. Levels of communication	1-2
1.2. PROFIBUS	1-3
1.3. Decentralisation	1-5
1.4. PROFIBUS protocol	1-6
1.5. Objectives of this manual	1-7
1.6. Using the manual	1-8
1.7. Further user information	1-9
 2. Safety and Reliability	 2-1
2.1. Target group	2-1
2.2. Reliability	2-1
2.3. Safety	2-3
2.3.1. To be observed during project planning and installation	2-3
2.3.2. To be observed during maintenance and servicing	2-4
2.3.3. Measures for the prevention of electrostatic charge	2-5
2.3.4. Security information for using the KUBES and VEBES software packages	2-6
 3. Hardware	 3-1
3.1. Description of the PROFIBUS equipment	3-2
3.1.1. PROFIBUS devices made by Kuhnke	3-2
3.1.2. The profibus cable	3-4

Table of contents

3.1.3. PROFIBUS connectors	3-5
3.1.4. The PROFIBUS interface on the device	3-5
3.1.5. Connecting junctions (T-junctions)	3-6
3.1.6. Bus termination	3-6
3.1.7. PC plug-in module	3-8
3.1.8. Repeaters	3-8
3.2. Setting up the PROFIBUS	3-9
3.2.1. Topology	3-9
3.2.2. Construction/Assembly	3-10
3.2.2.1. Cable Connection of the Stations	3-11
3.2.2.2. Bus termination	3-16
3.2.2.3. Grounding/EMC	3-18
3.2.2.4. Installation of a PROFIBUS PC plug-in board	3-19
3.3. Connecting Kuhnke devices to the PROFIBUS	3-20
3.3.1. KUAX 644 PC Control	3-20
3.3.1.1. Location of the PROFIBUS interfaces	3-20
3.3.1.2. Setting the transfer speed of the KUAX 644	3-21
3.3.1.3. Setting the PROFIBUS station address of the KUAX 644	3-22
3.3.2. KUAX 657P Modu Control	3-24
3.3.2.1. Location of the PROFIBUS interfaces on the module	3-24
3.3.2.2. Setting transfer speed and PROFIBUS station address	3-25
3.3.3. KUAX 680I Profi Control	3-27
3.3.3.1. Location of the PROFIBUS interfaces	3-27
3.3.3.2. Setting transfer speed and PROFIBUS station address	3-28
3.3.4. KUAX 680S Profi Control	3-30
3.3.4.1. Location of the PROFIBUS interfaces	3-30
3.3.4.2. Setting the PROFIBUS protocol and the PROFIBUS station address for the KUAX 680S	3-30
3.3.4.3. Transfer speed	3-32
3.3.5. KUAX 681M Profi Control	3-33
3.3.5.1. Location of the PROFIBUS interfaces	3-33
3.3.5.2. Setting transfer speed and PROFIBUS station address	3-34
3.3.6. Valve Island 799 PROFIBUS	3-36
3.3.6.1. Location of the PROFIBUS interfaces	3-36
3.3.6.2. Setting the PROFIBUS station address and the reaction of the outputs to system failures	3-37
3.3.6.3. Transfer speed	3-38

3.4. Programming of controllers via different interfaces	3-39
3.4.1. Programming via the V.24 interface	3-39
3.4.2. Programming via the PROFIBUS	3-41
3.4.3. Programming via the PC bus (KUAX 644)	3-41
 4. Software	 4-1
4.1. Programs	4-1
4.1.1. VEBES and KUBES: main purpose of the programs	4-1
4.2. Using VEBES to configure a PROFIBUS network	4-3
4.2.1. Creating a network	4-3
4.2.2. Setting the PROFIBUS parameters	4-4
4.2.3. Defining bus stations	4-6
4.2.4. Defining connections	4-8
4.2.4.1. Process chart communication connections	4-11
4.2.4.2. Block transfer communication connections	4-11
4.2.5. Using VEBES to integrate non-Kuhnke devices into the PROFIBUS network	4-13
4.2.5.1. Defining the device type	4-13
4.2.5.1.1. Object dictionary	4-14
4.2.5.1.2. Service Access Points	4-16
4.2.5.1.3. Process chart objects	4-17
4.2.5.1.4. Block transfer objects	4-18
Exercise: Defining devices	4-19
 4.3. Programming network projects with KUBES	 4-25
4.3.1. The Symbol Table	4-26
4.3.2. Addressing external operands in process chart communication .	4-27
4.3.2.1. Coding of operands	4-28
4.3.2.2. Addressing	4-29
Exercise: "Addressing operands in the Process Chart"	4-31
4.3.3. Programming block transfer (only PROFIBUS-FMS protocol)	4-35
4.3.3.1. KUBES modules for block transfer	4-38
4.3.3.2. Parameters of the KUBES modules	4-38
Exercise: "Cyclic block transfer" (single master network)	4-41
Exercise: "Block transfer on command" (single master system)	4-49

Table of contents

4.3.4. Connecting the programming PC with the controller 4-59

4.3.4.1. Online V.24 4-59

4.3.4.2. Online PC 4-59

4.3.4.3. Online PROFIBUS 4-60

4.3.5. Using KUBES to monitor the PROFIBUS network 4-60

4.3.5.1. Display address range 4-61

4.3.5.2. Display single address 4-61

4.3.5.3. Dynamic display in the Module Editor 4-61

4.3.5.4. Logic diagram 4-62

4.3.6. PROFIBUS messages 4-62

4.3.6.1. PROFIBUS error messages 4-63

4.3.6.2. PROFIBUS status messages 4-66

4.3.6.3. PROFIBUS event notifications 4-67

4.3.7. Monitoring the bus operation via the user program 4-69

Exercise: Bus control in the master 4-71

Appendices

A. A-1

B. PROFIBUS time behaviour B-1

C. List of manufacturers of PROFIBUS accessories C-1

Index Index-1

Sales & Service

Exercise program disk

1. Introduction

The increasing need for exchanging of information in highly rationalised firms is met today by the use of computer networks. They make optimal task management possible, because the information is received and processed in the place where it is entered but it will nevertheless be available at the place where it is needed later.

This type of network most commonly occurs in office environments, where they allow a quick flow of information between the buying, selling and bookkeeping departments, for example. Even the area of production is not excluded from an increasing incorporation in wide- ranging information systems.

The permanently increasing degree of automation in the field of industrial production technology makes a fast exchange of data necessary. To coordinate these fast processes of data exchange, hierarchically structured levels or layers of communication have been introduced.

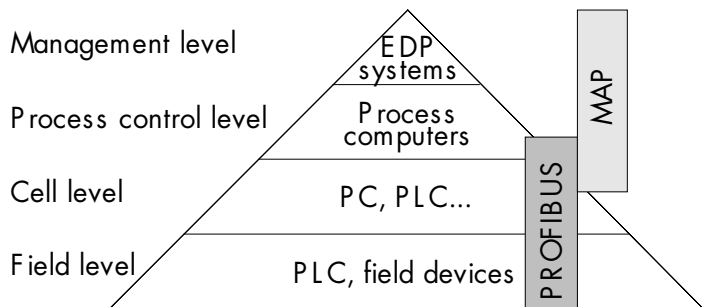
1.1. Levels of communication

There are four basic levels of communication:

- the management level which works with EDP systems;
- the process control level where central or process control computers are used for sub-configuration management, for example;
- the cell level for the processing of separate sub-tasks using PCs or programmable logic controllers (PLCs);
- the field level at which the actual technical process takes place. Here simple **field devices**, such as actuators and sensors are employed alongside programmable logic controllers.



Levels of communication



1.2. PROFIBUS

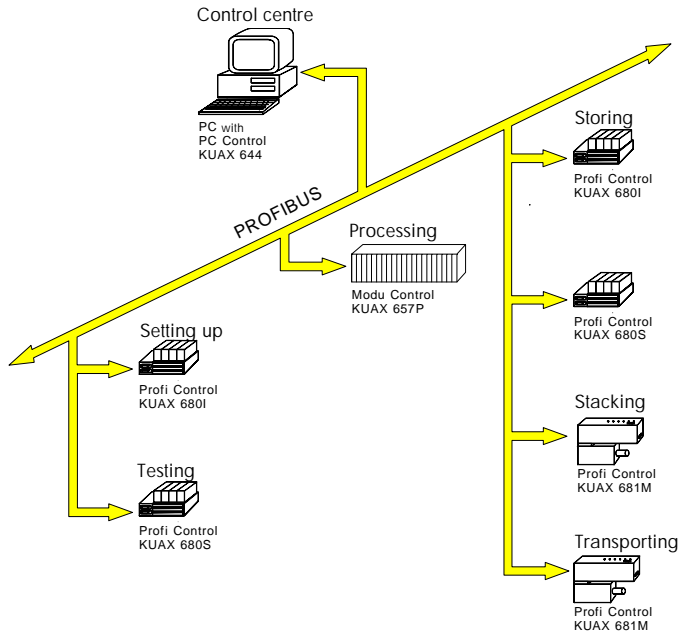
To achieve the highest efficiency of the information flow, all of these levels must be interconnected in networks and must allow a data exchange between levels. For the higher levels, this is guaranteed by the MAP (Manufacturing Automation Protocol) developed for the computer networks. MAP completely describes the communicative behaviour of the networking devices.

However, due to its complexity this system is not suitable for the field level. Communication at this level is based on fieldbuses, which work with a reduced communication protocol thus providing the speed that is necessary for monitoring and controlling the processes at the field level.

The PROFIBUS (Process Field Bus) is an example of this type of fieldbus. However, with its architecture and its connection to the cell level – and thus to the entire communication system – PROFIBUS is not limited to the restricted possibilities of a fieldbus.

PROFIBUS is an open fieldbus system. Open means that devices from various manufacturers can be integrated into the network to form a multi-vendor environment. This is made possible by the fact that the features of all devices which can be used on the PROFIBUS were standardized in DIN 19245 parts 1 and 2. This includes all inward and outward connections, interfaces, both parts of plug-and-socket type connectors but also all details of the communication protocol.

Additionally, application-orientated profile classes were – and will continue to be – developed making a dedicated use of the available devices possible.



As with the MAP, the basis of the PROFIBUS protocol is the communication architecture of the **ISO/OSI reference model**, which is explained in more detail in the glossary.

PROFIBUS covers the field level and part of the cell level of the communication hierarchy, and is appropriate means for the connection of simple **field devices** and controllers such as programmable logic controllers.

1.3. Decentralisation

PROFIBUS allows decentralising work routines. Like this every subsystem is used where it is needed and carries out the tasks for which it is optimally appropriate.

Decentralisation has the following advantages:

- cost reduction by economizing on multicore cables, space and assembly time;
- the program structure corresponds to the object structure which makes the whole system clearer;
- easier trouble-shooting because the overall process is divided into clear sections;
- facilitates servicing: it is no longer necessary to shutdown the entire plant for maintenance work but rather single devices can be disconnected (modular machines);
- reduced commissioning times;
- program and function tests can be carried out in individual stations prior to implementing a complete solution.

Moreover, networking via PROFIBUS allows you to:

- combine devices from different manufacturers;
- collect data centrally;
- collect process data directly;
- keep error statistics without any trouble;
- control complex processes with many devices;
- divide complex tasks between several processors.

1.4. PROFIBUS protocol



Kuhnke provide PROFIBUS in two different protocol versions: as the PROFIBUS-FMS **protocol** (Field Message Specification) and as the SINEC L2 -DP **protocol** (Decentralized Periphery). PROFIBUS-FMS corresponds to the fieldbus providing the full set of functions as described in every detail in DIN 19245, Part 1. Amongst other things, this protocol provides for operating multi-master systems with a comparatively large extent of data communication.

The other protocol, SINEC L2-DP, is described in DIN 19245, Part 2. It was developed as a communication solution for one **master** and up to 124 **slaves** representing a slightly different architecture to the FMS-protocol (see glossary). Compared to the latter, SINEC L2-DP works with a limited set of functions. Thus, while only mono-master systems can be operated for example, a bus running the SINEC L2-DP protocol also has much shorter reaction times as the data exchange between any 2 bus stations is also reduced.

The following table gives an overview of the most important differences between the two PROFIBUS protocols.

PROFIBUS-FMS	SINEC L2-DP
multi-master system	mono-master system
process map and block transfer possible	only process map possible
address of master: optional up to address #126*	address of master: 2 or 1
address of slave: optional up to address #126*	address of slave: 3 through to 126*
DP station can be implemented in the network	FMS station cannot be implemented in the network
programmable via PROFIBUS	not programmable via PROFIBUS

* only up to 99 under VEBES

The hardware interconnection method of the individual devices is the same for both systems, i.e. interfaces, cables and connectors are the same in both cases.

It entirely depends on your application to give preference to one of the two systems.

1.5. Objectives of this manual

This instruction manual aims at making the integration of Kuhnke PLCs and other Kuhnke PROFIBUS devices into the PROFIBUS environment easier for the planners, project planning engineers and users. It contains information about the necessary material, the installation of PROFIBUS, the necessary Kuhnke software and its use in PROFIBUS. Using easy examples, the user will be guided through the required steps. The documentation is supplemented by an appendix, which contains a list of manufacturers and a part about the behaviour of the PROFIBUS over time as well as a detailed glossary providing the user with a brief introduction into PROFIBUS terminology.

1.6. Using the manual

Basically, you should read the whole manual (with the exception, perhaps, of the glossary). The chapter "Safety" is particularly important. Instructions are given here which should be taken note of for safe operation of PROFIBUS.

Part 3, "Hardware", contains information which is relevant to specific Kuhnke devices. Thus it is easy to find the information which is important for your network structure. The same is valid for chapter "PROFIBUS messages" in the software section.

You will repeatedly find symbols in the margin which are meant to draw your attention to certain types of information thus providing convenient means for 'navigating' through the manual.



Warning. Here you will find important safety information which you should take note of (see chapter "Safety").



Dangerous Voltage. This symbol warns you of dangers which could cause death, (grievous) bodily harm or material damage if the described safety measures are not taken.



Hand. This marks cross references or references to additional information.



Book. On pages where the book appears, you will find terms in bold print. Detailed explanations of these terms are given in the glossary.



Mouse. This appears in the margin of the exercises whenever you are requested to carry out mouse operations on the computer.



Key. This appears in the margin of the exercises whenever you are requested to make entries via your computer keyboard.

1.7. Further user information

Before you start putting the PROFIBUS network configuration into operation, we strongly recommend also referring to the operating instructions for the individual Kuhnke devices which are to be integrated as well as this user manual. These instruction manuals are delivered with the corresponding devices but you may also obtain them separately from Kuhnke. Please consider the following manuals:

PC Control KUAX 644	E 331
Modu Control KUAX 657P	E 312
Profi Control KUAX 680I	E 308
Profi Control KUAX 680S	E 307
Profi Control KUAX 681M	E 311

There is a whole range of Beginner's Manuals available for the necessary Kuhnke software, including KUBES, the programming software, VEBES, the network configurator software, and, if applicable, GrafKEd, the graphical Kuhnke editor for sequential function charts. The beginner's manuals explain the program installation and guide you through the first steps into the programs. A detailed documentation of the software packages comes to you in the shape of context-sensitive online help systems.

Beginner's manuals:

KUBES	E 327
VEBES	E 315

2. Safety and Reliability

2.1. Target group

This instruction manual contains all information necessary for the use of the described product (control device, software, etc.) according to instructions. It is written for the **personnel of the construction, project planning, service and commissioning departments**. For proper understanding and error-free application of technical descriptions, instructions for use and particularly of notes of danger and warning, **extensive knowledge of automation technology** is compulsory.

2.2. Reliability

Reliability of Kuhnke controllers is brought to the highest possible standards by extensive and cost-effective means in their design and manufacture.

These include:

- selecting high-quality components,
- quality arrangements with our sub-suppliers,
- measures for the prevention of static charge during the handling of MOS circuits,
- Worst-Case dimensioning of all circuits,
- inspections during various stages of fabrication,
- computer aided tests of all assembly groups and their efficiency in the circuit,
- statistic assessment of the quality of fabrication and of all returned goods for immediate taking of adjustment measures.

Safety and Reliability

Despite these measures, the occurrence of failures in electronic control units - even if most highly improbable - must be taken into consideration.

All KUHNKE software products, too, undergo extensive quality assuring procedures during their development.

These include:

- software specification
- rough and detailed design
- extensive documentation of the development steps
- testing of individual software modules
- integration test of the software package
- application test software « controllers
- integration into the KUHNKE Quality Assurance system
- transferring the finished software to error-free installation disks.

Despite these measures, we cannot warrant a perfect functioning of our software, as there may be unexpected hardware configurations, for example, which might cause unforeseeable malfunctions.

2.3. Safety

Our product normally becomes part of larger systems or installations. The following notes are intended to help integrating the product into its environment without dangers for man or material/equipment.

2.3.1. To be observed during project planning and installation



To achieve a high degree of conceptual safety in planning and installing an electronic control unit it is essential to follow the instructions given in the manual exactly because wrong handling could lead to rendering measures against dangerous failures ineffective or to creating additional dangers.

Electrically (24V DC power supply):

- provide sufficient separation of low voltage;
- apply power packs in accordance with IEC 364-4-41 or CENELEC HD 384.4.41 (VDE 0100, Part 410) respectively;
- in case of power breakdowns or power fades: the program has to be structured in such a way as to create a defined state at restart that excludes dangerous states;
- emergency switches or other emergency installations have to be realized in accordance with EN 60204/IEC 204 (VDE 0113). They must be effective at any time;
- safety and precautions regulations for qualified applications have to be observed;
- please pay particular attention to the notes of warning (see Introduction, Symbols) which, at relevant places, will make you aware of possible sources of errors;
- the relevant standards and VDE regulations are to be observed in every case;
- install control elements in such a way as to exclude unintended operation;
- lay control cables in such a way as to exclude interference (inductive or capacitive) which could influence the operation of the controller.

Pneumatically:

- Make sure that all regulations relevant for the working with pressurized air are taken into consideration at any time during installation and putting into operation of the control devices;
- observe the security information given in the appendix of every pneumatics main catalogue to guarantee trouble-free operation of the components;
- the pneumatic pressure system must be constructed in a way that excludes uncontrolled operations of pneumatic elements in case of the pressurized air supply breaking down or fading and that guarantees a defined and non-dangerous condition of the system at restart after breakdown or fading of the air supply.

2.3.2. To be observed during maintenance and servicing



Electrically (24 V DC power supply):

- during measuring and checking operations on a controller in a power-up condition, precaution regulation VBG 4.0 must be observed and §8 (Admissible deviations during working on parts) in particular;
- repairs must only be executed by the trained Kuhnke personnel (usually in the main factory in Malente). Warranty expires in every other case;
- spare parts:
 - only use parts approved of by Kuhnke. Only genuine Kuhnke modules (OEM) must be used in modular controllers;
- modules must only be connected to or disconnected from the controller with no voltage supplied. Otherwise they may be destroyed or (possibly not immediately recognizably!) detracted from their proper functioning;
- always deposit batteries and accumulators as hazardous waste.



All warranty expires if you open the electronics module.

Pneumatically:

- Defective valves may be exchanged by trained service personnel if the applicable regulations for the prevention of accidents are observed;
- make sure that the valves are exchanged with no power or pressure applied;
- the relevant assembly and service instructions must be observed to guarantee proper functioning.

2.3.3. Measures for the prevention of electrostatic charge

Electrostatic charge is dangerous for components and assembly groups. It is a peculiarity of electrostatics to not destroy the sensitive components but to damage them in a not immediately conceivable way. It is because of this that devices stop functioning after some time of service.

The ESD measures (ESD = electrostatic discharge) executed in the factory are only guaranteed to be effective if they are also regarded by the user (service).



Please note:

- *only store parts in their factory-packing or in an antistatic packing of similar quality;*
- *assembly groups must only be touched by persons who are grounded via a wrist bracelet and/or a discharging mat and shoe-grounding strips (observe protection of people!);*
- *only ship assembly groups in their factory-packing or in an antistatic packing of similar quality.*

Reference to literature (3M Deutschland GmbH, Neuss):

Information brochure

"Wissenswertes über die Elektrostatik in der Mikroelektronik"
(Interesting Facts about Electrostatics in Microelectronics)

2.3.4. Security information for using the KUBES and VEBES software packages

KUBES and VEBES permanently monitor your entries and:

- prompt you with safety messages if you make any dangerous entries. You will have to confirm these messages by clicking on OK. This is to avoid erroneous entries only due to negligence that might lead to dangerous situations;
- report invalid user entries during programming. The online help systems contain danger messages in obvious boxes. In this manual, warning texts are marked by a warning triangle (see chapter Introduction, Symbols).

We would still like to **again** mention some possible causes for danger:

KUBES:

- **Program design:** All controller programs must, by their design, exclude dangerous states after voltage cuts by guaranteeing a defined status after restart.
- **External Operands:** If you are embedding external operands in your program code (e.g. inputs of a KUAX 680S) make sure that you are working with valid operands. If, for example, the bus connection has been interrupted for whatever reason, the value received does no longer correspond to the actual status of the input. This may result in faulty switching conditions in the circuit. To avoid these we strongly recommend always analysing the PROFIBUS messages carefully in the program.
- **Input addresses:** When reading input addresses you must make sure only to use addresses which do actually exist. You might be working with non-existing addresses, for example, if you are addressing an unplugged module in an external I/O device or if the PROFIBUS connection to this device is interrupted.

- **Option Start <RUN>, PLC menu, Main menu bar:** Make sure that your program is error-free before you start it with a machine connected to the controller. Controlling the operation of any installation via a device with a faulty program might cause damage to man and machine. KUBES provides you with extensive test functions to debug your program prior to putting it into operation.
- **Option Stop, PLC menu, Main menu bar:** Only use this option during test operation with no machine connected to the controller. The status of outputs and markers remains unchanged when you stop the program. So, if a machine is connected to the controller then this might lead to dangers for man and machine because you can no longer influence activated outputs via the program. We strongly advise you to always using the "Stop and Reset" command if there is a machine connected to the controller.
- **Display single address and Display address range, PLC menu, Main menu bar:** Both display options give you the possibility to overwrite operand values with preset values. Under certain circumstances, this might lead to dangers for man and machine (e.g. a motor starts, a steam valve opens, etc.).

If a machine is connected to your controller, always make sure that such dangers are excluded before you transmit preset values to the controller.

- **Dynamic displays of the Display single address, the Display address range, and in the PLC menu of the Module editor:** The dynamic display depends on data being read from the controller and displayed on the screen. Every data exchange causes additional load on the capacity of the controller CPU; the cycle time will thus be extended. The dynamic display may therefore lead to malfunctions in controllers with particularly fast processes.

VEBES:

- **Network menu, option Bus parameters:** If you make wrong entries into the time parameter fields you may render bus operation impossible altogether.
- **Network menu, option Station:** If you delete a master you will also delete the corresponding network project.
- **Network menu, option Station:** The configuration of a slave entered into the program must correspond to the actual configuration of the device.

3. Hardware



The **project planning** of a network like PROFIBUS always follows the same basic steps:

- dividing the whole process to be controlled into separate sub-tasks;
- defining the number and type of devices required;
- establishing the network layout and the network topology;
- acquiring the needed equipment;
- setting up the network;
- connecting the devices;
- configuring the network (this is described in the software section);
- programming the devices;
- putting the network into operation.

The following chapter contains all information you need for the mechanical setting-up of a PROFIBUS network with Kuhnke devices. It is divided into 4 parts which aim at giving you an easy overview of the PROFIBUS structure starting at the first planning phase:

Part 1 describes the equipment you need

Part 2 contains general information for installation

Part 3 documents the connection of individual Kuhnke devices into the PROFIBUS

Part 4 presents the programming possibilities using different interfaces

3.1. Description of the PROFIBUS equipment

Apart from the devices you want to run on your PROFIBUS network, you will need the following hardware to make the PROFIBUS:

- PROFIBUS cable
- connector plugs
- PROFIBUS interface on the device
- connecting junctions (T-junctions)
- bus termination connector
- active bus termination connector
- PROFIBUS PC-board, if required
- repeater, if required
- bus monitor, if required

3.1.1. PROFIBUS devices made by Kuhnke

Kuhnke have a whole range of devices which can be used on the PROFIBUS. These include the programmable logic controllers KUAX 644 PC Control, the KUAX 657P Modu Control (if fitted with a PROFIBUS module), and the KUAX 680I Profi Control. Other types of devices are the KUAX 680S Profi Control, a decentralized input and output unit, the KUAX 681M Profi Control, a decentralized positioning controller, and the Valve Island 799 which can also be operated on the PROFIBUS. From now on, these devices will only be referred to by KUAX plus type number.



The PROFIBUS User Organization (PNO) defines guidelines as supplements to the PROFIBUS standard. These guidelines are the basis for developing so-called **profile classes** whose purpose it is to classify types of PROFIBUS devices for specific branches of industry and their needs. A profile class is a collection of characteristics which determine the behaviour of the corresponding device in PROFIBUS communication and towards the application.



One of the things specified by a profile class is whether a device is defined as a client (**master**; service requester) or a server (**slave**; service provider). The actual signal processing is done by the higher-ranking controllers, or **masters**. These have the profile class of a controller. Kuhnke devices which work as servers, or **slaves**, have the sensor/actuator profile class 2. The following table is an overview of the Kuhnke devices and their profile classes.

PROFIBUS devices by Kuhnke and their profile classes

Device	Device type	Profile class	Function	PROFIBUS protocol	
				FMS	DP
KUAX 644	programmable logic controller	controller profile class 2	client	yes	yes
KUAX 657P (with PROFIBUS module)	programmable logic controller	controller profile class 2	client	yes	yes
KUAX 680I	programmable logic controller	controller profile class 2	client	yes	yes
KUAX 680S	input and output device	sensor/actuator profile class 2	server	yes	yes
KUAX 681M	decentralized positioning controller	sensor/actuator profile class 2	server	yes	no
Valve Island 799	valve island	sensor/actuator profile class 2	server	yes	no

3.1.2. The PROFIBUS cable

The cable which can be used in PROFIBUS has the following characteristics:

- shielded twisted pair
- cross section $\geq 0.22 \text{ mm}^2$ (using a cross section of $\geq 0.5 \text{ mm}^2$ doubles the maximum length of the cable)
- characteristic impedance 100 to 130Ω
- cable capacity approx. 60 pF/m
- length of entire cable: the max. permissible length of the cable is reversely proportional to the used data transmission speed (baudrate).

Interrelation between length of cable and transmission speed

Baud rate [kBaud]	Max. line length [m]
500	200 (400)
19.2	1200 (2400)
9.6	1200 (2400)



The cable lengths indicated in the table above are valid for a cross sections of $\geq 0.22 \text{ mm}^2$. For cross sections $\geq 0.5 \text{ mm}^2$ refer to the numbers in brackets. "Length" stands for the length of the entire **bus**, or up to the first repeater.



Please see the appendix for a list of manufacturers and order numbers.

There are highly flexible cables and cables with special jackets available for cables laid in the ground or cables used on permanently moving machine parts.

3.1.3. PROFIBUS connectors

Three different types of connector are used for Kuhnke devices. For the KUAX 644, the PROFIBUS module of the KUAX 657P and the KUAX 680S, 9pin D-SUB connectors are used. The KUAX 681M is connected with 4pin round connectors and the Valve Island 799 with 6pin ones. See the appendix for a list of manufacturers.

3.1.4. The PROFIBUS interface on the device

All devices to be connected to the PROFIBUS must have a PROFIBUS interface. The interface used is a potential separated RS485 interface. The following Kuhnke devices are equipped with one or two PROFIBUS connector sockets:

Kuhnke devices with 1 PROFIBUS connector socket	Kuhnke devices with 2 PROFIBUS connector sockets
KUAX 644 KUAX 680I**	PROFIBUS module of the KUAX 657P* KUAX 680I** KUAX 680S KUAX 681M Valve Island 799

* The KUAX 657P can be retrofitted with RS485 interfaces using a PROFIBUS module (see appendix).

** The KUAX 680I is available with either 2 RS485 interfaces or 1 RS232 (V.24) interface.

In cases where the device is equipped with two PROFIBUS interfaces, these are connected in parallel, i.e. they have the same function.

3.1.5. Connecting junctions (T-junctions)

T-junctions are used to connect spur lines that branch from the PROFIBUS cable to individual devices. T-junctions are required for devices with only one PROFIBUS connector. Spur lines have a max. length of 30 cm.

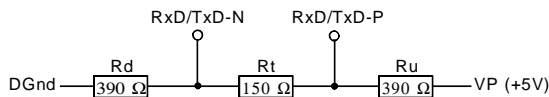
For the KUAX 680S, KUAX 681M and the Valve Island 799 no spur lines are required because they have 2 PROFIBUS interfaces. All devices with 2 interfaces can be directly connected to the PROFIBUS network (see chapter "Connecting Kuhnke devices to the PROFIBUS").



It might be practical to also connect devices with 2 PROFIBUS interfaces by using a T-junction. In this case the bus will not be interrupted when adding on the device.

3.1.6. Bus termination

Both ends of the PROFIBUS cable (not the spur lines!) must be connected to a line termination resistor. Furthermore, a pull-down resistor is to be put between RxD/TxD-N and the data reference potential (Gnd) and a pull-up resistor between RxD/TxD-P and the supply voltage. This defines an idle potential in the cable when there is no data communication.



There are two possibilities:

- A terminating station has 2 PROFIBUS connections which are connected in parallel (see table on page 3-5).

In this case, a pre-assembled Kuhnke bus termination connector already fitted with the necessary resistors, can be plugged into the remaining connector position (see appendix, Accessories). The bus termination connectors are always available in pairs, one male and one female.



The bus termination must always be supplied with voltage. Should it be possible to switch off a terminating station during bus operation, then you must use a bus termination with its own supply (active bus termination supplied with 230V AC, see appendix).

- The terminating station has only one PROFIBUS connection (see table on page 3-5).

In this case you must fit the connector with resistors. This process is described on page 3-17.

The KUAX 681M is equipped with bus termination resistors when it comes to you. The pull-up and pull-down resistors are of very high impedance and permanently connected - i.e. causing no problems.

The actual bus termination is internally connected to the RxD/TxD-N line on one side and to pin 4 on the other. If the KUAX 681M is a terminating station, then the bus termination need only be activated. This is explained on page 3-17.

In the Valve Island 799, too, all you need to do is to put in two jumper wires to activate the built-in bus termination (see page 3-18).

3.1.7. PC plug-in module

If you want to use the PROFIBUS interface to program a controller, e.g. a KUAX 680I, then your PC must be equipped with a PROFIBUS interface.

We recommend using the PC interface card provided by TMG i-tec (see appendix) as a PC plug-in module. The PROFIBUS driver for KUBES was developed for this interface card. The data format is laid down by PROFIBUS norm DIN 19245.

Please refer to pager 3-19 for notes on using a PC interface card.

3.1.8. Repeaters



Repeaters allow the bridging of long lengths of cable and the connection of a larger number of stations on the **bus**. Furthermore, branched network topologies can be realized with these.

We recommend using regenerating repeaters, which provide a signal refresh feature. Examples for regenerating repeaters are the "RS485 Repeater" or the "Power Repeater" (for industrial environments with particularly high interference) provided by the Wieland company (see appendix).

3.2. Setting up the PROFIBUS

3.2.1. Topology



What we mean by topology in this case is the architecture of a network. The PROFIBUS is constructed as a line. The number of stations on the bus is limited to 32. If this is not sufficient you can add another line which is connected to the first line by a bi-directional line amplifier (repeater). A repeater counts as an interface on the **bus**, thus reducing the maximum number of stations. Thus the number of "real" stations on one line is reduced to 31 (or to 30 if 2 repeaters are used). However, repeaters do not receive a PROFIBUS station address for network configuration.

Using repeaters can thus increase the number of bus stations to 122. This is assuming that you are working with non-regenerating repeaters which provide no signal refresh feature, in which case a max. number of 3 repeaters between two stations is allowed.

With the help of repeaters, larger distances can be bridged independently from the number of connected stations. The permissible maximum length of the PROFIBUS cable otherwise entirely depends on the set transmission speed (see page 3-4).

If you connect the lines by regenerating repeaters, you can also implement tree or star structures. To achieve this different kind of topology you use the repeaters not only as connecting points but also as branching junctions. Like this you can work with the maximum number of bus stations which number is only limited by the address limit of 126.

Please obtain information about the operative possibilities and the connection of repeaters from the corresponding supplier.

3.2.2. Construction/Assembly



The order of the stations on the **bus** is not important. Each station is addressed individually by its individual PROFIBUS station code.

Assembly of the PROFIBUS is easy:

- connect the PROFIBUS cable to the RS-485 interace(s) of all stations;
- use repeaters to establish further lines if required;
- plug a bus termination connector into the starting and terminating device of the PROFIBUS;
- use the DIP switches on the devices to set the PROFIBUS station address of every individual device;
- use DIP switches to set the bus data transmission speed (baud rate);
- install a PC plug-in module (KUAX 644) if required

This section describes the general steps that have to be taken for setting up the PROFIBUS. In the next section you will find notes on how to connect the different Kuhnke devices to the PROFIBUS.

3.2.2.1. Cable Connection of the Stations

KUAX 644
KUAX 680I
KUAX 657P
KUAX 680S

The PROFIBUS connectors of KUAX 644, KUAX 680I and the PROFIBUS modules of KUAX 657P and KUAX 680S are 9-pin D-SUB plugs with the following pin assignment:























Assignment of 9-pin D-SUB plug

Pin	Signal	Description
1	Shield	shield or ground (also connected to plug casing)
2		
3	RxD/TxD-P	receive/transmit data-P
4		
5	DGnd	data reference potential (Ground)*
6	VP	plus pole of supply voltage (+5V)*
7		
8	RxD/TxD-N	receive/transmit data-N
9		

* DGnd and VP are necessary for bus termination. For PROFIBUS stations only pins 3 and 8 are needed.

Hardware

The two data lines, RxD/TxD-P (pin 3) and RxD/TxD-N (pin 8), are connected 1:1. Crossover connections are not allowed. We recommend always connecting the lighter wire to RxD/TxD-P and the darker one to RxD/TxD-N:

		D-Sub, 9pol.	Bus cable		Partner
					
Shield		1			Shield
		2			
RxD/TxD-P		3			RxD/TxD-P
		4			
DGnd		5			DGnd
VP		6			VP
		7			
RxD/TxD-N		8			RxD/TxD-N
		9			

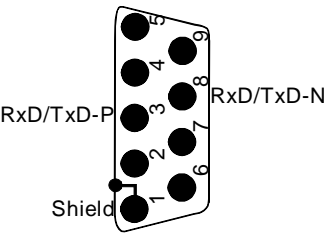
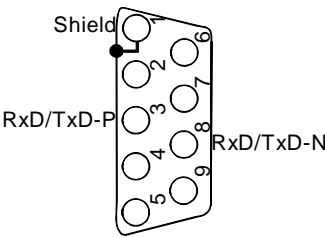


In Kuhnke devices, the connection between Pin 1 and the connector housing is conductive. Thus the shielding can also be connected to the connector housing.

Contact layout

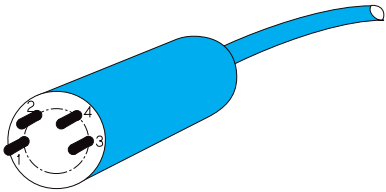
Socket

Plug



KUAX 681M The KUAX 681M is connected to the PROFIBUS with 4-pin connectors:

Contact layout of the cable plugs



Cable plug

4-pin connector

		Pin	Signal	Description
wire bridge		1	RxD/TxD-N	receive/transmit data-N
		2	RxD/TxD-P	receive/transmit data-P
		3	Shield	shield or protective ground
		4	Rt active	activate bus termination *)

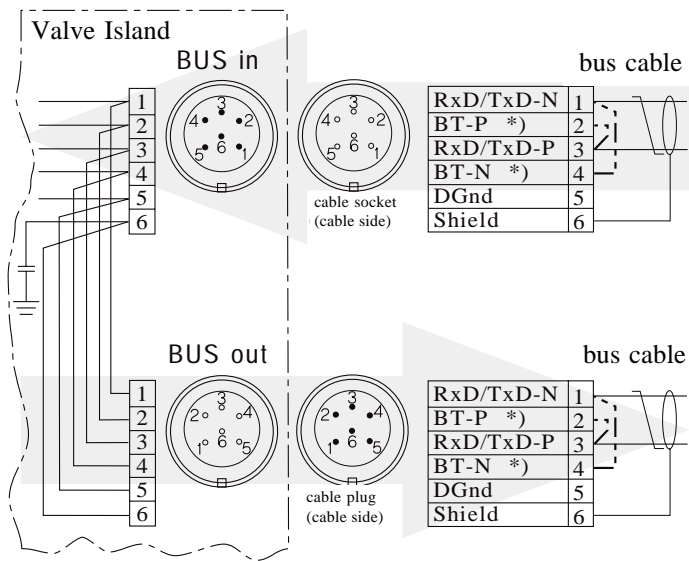
*) only if terminating station: bridge pins 2 and 4

The two data lines, RxD/TxD-P (pin 2) and RxD/TxD-N (pin 1), are connected 1:1. Crossover connections are not allowed. We recommend always connecting the lighter wire to RxD/TxD-P and the darker one to RxD/TxD-N.

Hardware

Valve island
799

The Valve Island 799 is equipped with 6-contact connectors:



6-pin connector

Pin	Signal	Description
1	RxD/TxD-N	receive/transmit data-N
2	BT-P*	activate bus termination
3	RxD/TxD-P*	receive/transmit data-P
4	BT-N*	activate bus termination
5	DGnd	data reference potential(ground)
6	Shield	shield or protective ground

wire bridges

* Only if terminating station: bridge pins 1 and 4, and pins 2 and 3 as bus termination.

The two data lines, RxD/TxD-P (pin 3) and RxD/TxD-N (pin 1), are connected 1:1. Crossover connections are not allowed. We recommend always connecting the lighter wire to RxD/TxD-P and the darker one to RxD/TxD-N.



The stations can be connected to the **bus** after the cable has been equipped with the corresponding connectors. There are 2 possibilities:

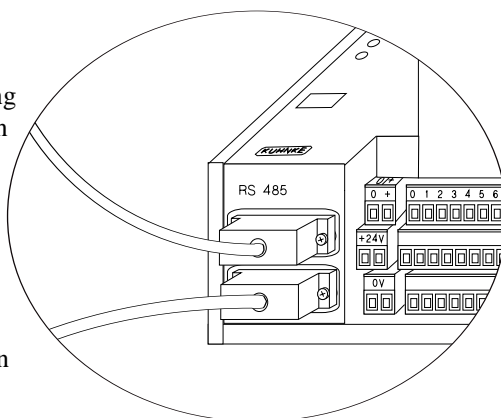
KUAX 680I
KUAX 657P
KUAX 680S
KUAX 681M

Stations with 2 parallel interfaces, type RS485 (PROFIBUS module for the KUAX 657P, KUAX 680I, KUAX 680S, KUAX 681M):

The interfaces in the device are usually arranged underneath each other. Most stations use 9-connector D-SUB plug-and-socket connectors (exceptions are the KUAX 681M and the Valve Island). In these cases, the upper connector has pins and the lower one has sockets (exception: PROFIBUS module of KUAX 657P where the connectors are arranged the other way around). Connect the cable coming in from the preceding PROFIBUS station to the male connector, and connect the cable going out to the subsequent bus station to the female connector.

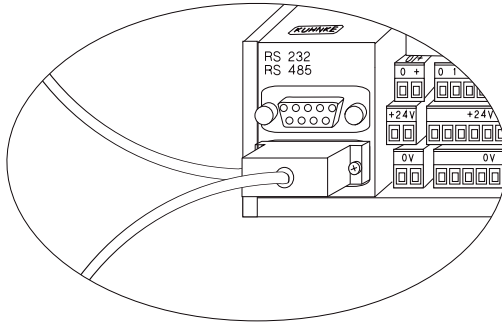
from the preceding
PROFIBUS station

to the subsequent
PROFIBUS station



PROFIBUS stations with only one interface RS-485:

These stations are connected to the PROFIBUS by a T-junction. The spur lines must be no longer than 30 cm.



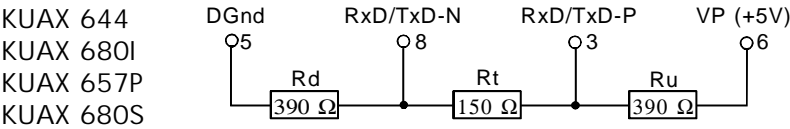
Even when the device is turned off, the connection between the preceding and subsequent stations remains intact as long as you leave the connectors plugged in. Unplugged connectors can be connected to one another to maintain the bus connection between the other stations.

3.2.2.2. Bus termination

Both ends of the PROFIBUS cable must be connected to resistors to ensure a defined idle potential (see page 3-6).

For terminating partners equipped with 2 PROFIBUS connectors you can simply plug in a pre-assembled bus termination connector to the unused connection.

The user of terminating partners with only one PROFIBUS connection must make the following pin assignment in the connector:



Connect the resistors to the pins before using the plug to connect the terminating partner to the **bus**.



Never use a two-core cable to connect the bus termination as the power supply of the resistors must be guaranteed (VP and DGND, see illustration).

KUAX 681M

The KUAX 681M and the Valve Island 799 are already equipped with bus terminating resistors. If they are terminating partners, i.e. only connected to one other station, then the bus termination on the KUAX 681 M is activated by a wire bridge (wire thickness as for the PROFIBUS cable) between pins 2 and 4 of the connector:

	Pin	Signal	Description
	1	RxD/TxD-N	receive/transmit data-N
	2	RxD/TxD-P	receive/transmit data-P
	3	Shield	shield or protective ground
wire bridges	4	Rt active	activate bus termination *)

*) only if terminating station: bridge pins 2 and 4

Hardware

Valve Island
799

The built-in bus termination of the Valve Island 799 must also be activated on the unused bus connector. For this purpose, put a wire bridge between pins 1 and 4 and another one between pins 2 and 3 (wire thickness as for the PROFIBUS cable):

wire
bridges

Pin	Signal	Description
1	RxD/TxD-N	receive/transmit data-N
2	BT-P*	activate bus termination
3	RxD/TxD-P*	receive/transmit data-P
4	BT-N*	activate bus termination
5	DGnd	data reference potential(ground)
6	Shield	shield or protective ground



The terminating partners must always remain switched on to maintain the idle potential of the PROFIBUS cable. They may only be switched off if an "active" bus termination with its own power supply has been used (see page 3-6).

3.2.2.3. Grounding/EMC

For reasons of operational safety, connect the shielding to the location provided on the device (see the instructions manual of the corresponding supplier).

In Kuhnke devices, the 0-V lines of the controllers and the connection for the shielding of the bus cable are connected to the casing ground by capacitors. This bleeds off high frequency disturbances.

Make sure to ground-connect the basic devices to render these measures effective. Please refer to the instruction manual of your controller for a description of how to connect the grounding wire.

3.2.2.4. Installation of a PROFIBUS PC plug-in board

If you want to program a controller via the PROFIBUS interface, then the corresponding PC must be equipped with a PROFIBUS interface. In this case, you use a PROFIBUS cable to connect the PROFIBUS interface of the computer to the controller. The computer thus becomes a separate PROFIBUS station.



Programming via the PROFIBUS interface is only possible when using the PROFIBUS-FMS protocol.

Install the interface board according to the manufacturer's instructions.

It contains a dual-port RAM. Use the jumper strip on the board to set the DP-RAM addresses of the PC address area for the PCB. The PC needs to know these addresses. Make the following entry in WIN.INI (data file containing the WINDOWS environment and startup parameters, see your WINDOWS manual). You must make this entry manually as it is not made automatically during installation:

[KUBES]

PROFIBUSADR=0xXXXX*

Where XXXX is the PROFIBUS station address to which the PROFIBUS module has been set.

E.G. XXXX=C800 for address C800h.

* The [KUBES] section already exists in your WIN.INI if KUBES has been installed on the PC. The line "PROFIBUSADR=0xXXXX" is the information you are obliged to enter manually to complete installation.

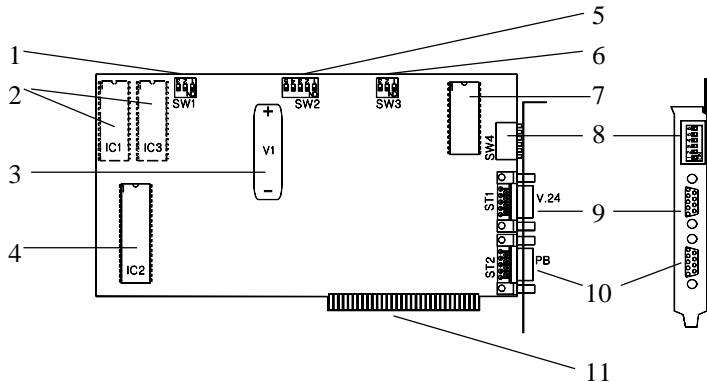
Please refer to the description of your PROFIBUS interface card for possible changes to your CONFIG.SYS and further information concerning this topic.

3.3. Connecting Kuhnke devices to the PROFIBUS

3.3.1. KUAX 644 PC Control

3.3.1.1. Location of the PROFIBUS interfaces

The KUAX 644 has 2 interfaces which are located on the front side of the module. The upper one is a V.24 interface (RS232) for programming via the PC and the lower one is the PROFIBUS interface (RS485).



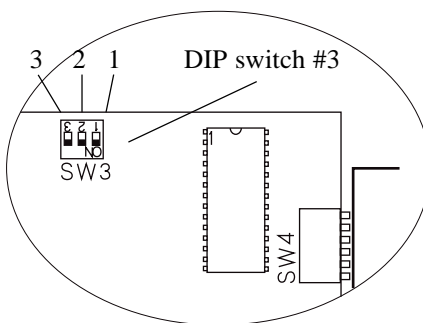
- 1 DIP-switch #1, type of storage
- 2 user memory (Flash) ERROM
- 3 accumulator
- 4 PLC monitor program
- 5 DIP-switch #2, adaptation to PC
- 6 DIP-switch #3, PROFIBUS transfer rate
- 7 PROFIBUS monitor program
- 8 DIP-switch #4, PROFIBUS station address
- 9 V.24 interface
- 10 PROFIBUS interface
- 11 PC-bus interface

The KUAX 644 is connected to the PROFIBUS by a T-junction. The connection to the PROFIBUS is described on page 3-11.

3.3.1.2. Setting the transfer speed of the KUAX 644

The PROFIBUS interface is controlled by a 8051 processor, which is located on the right at the top of the printed circuit board with the appropriate monitor EPROM.

In this area there are 2 DIP switch blocks. These serve to set the transfer speed (baudrate) and the PROFIBUS station address. The 3-pin DIP-switch block #3 serves to set the baudrate. It is located on the right and at the top of the board.



The following table contains the DIP switch settings for the various PROFIBUS transfer rates:

1	2	3	Baudrate[KBaud]	Max. line length [m]
off	off		500	200
on	off		19.2	1200
off	on		9.6	1200

3.3.1.3. Setting the PROFIBUS station address of the KUAX 644

At its front side, the KUAX 644 is equipped with DIP-switch block (#4) which is accessible from outside. Use this for setting the PROFIBUS station address of your KUAX 644. The following values are assigned to the DIP-switch settings:

Dip switch								PROFIBUS station address
1	2	3	4	5	6			
off	off	off	off	off	off			0
on	off	off	off	off	off			1
off	on	off	off	off	off			2
on	on	off	off	off	off			3
off	off	on	off	off	off			4
on	off	on	off	off	off			5
off	on	on	off	off	off			6
on	on	on	off	off	off			7
etc. through to:								
off	on	on	on	on	on			62
on	on	on	on	on	on			63
1	2	4	8	16	32	←	significance of switch 1-6 (binary coded)	



The settings of the PROFIBUS encoding switch are only read during the PC startup sequence. Changes during the operation will not be accepted.

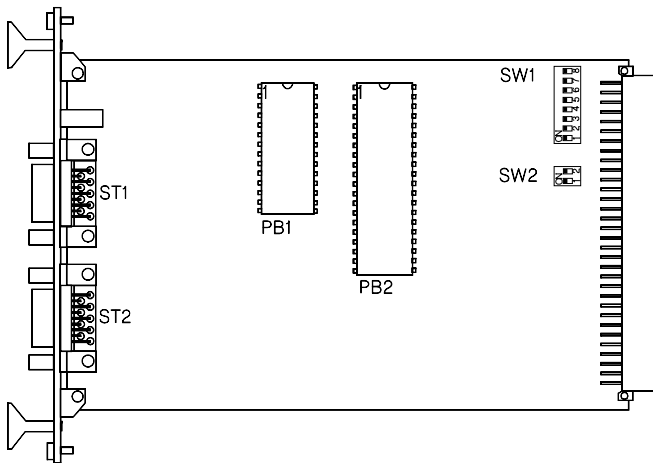
3.3.2. KUAX 657P Modu Control

You can only use the KUAX 657P on the PROFIBUS if the device has been equipped with a PROFIBUS module (see appendix).

3.3.2.1. Location of the PROFIBUS interfaces on the module

Both of the PROFIBUS interfaces are located on the front side of the PROFIBUS module.

The connection to the PROFIBUS is described on page 3-11.



ST1: female bus connector

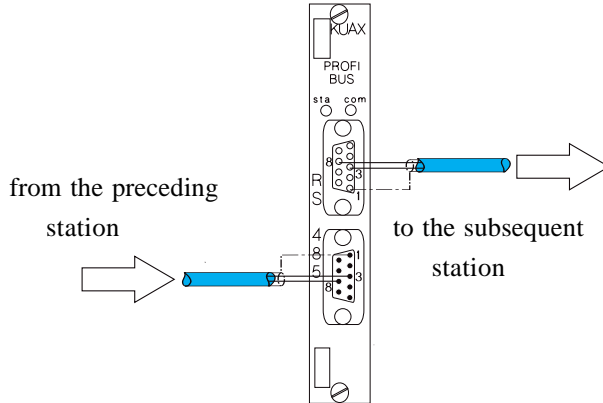
ST2: male bus connector

PB1: EPROM with PROFIBUS application layer software

PB2: EPROM with ALI

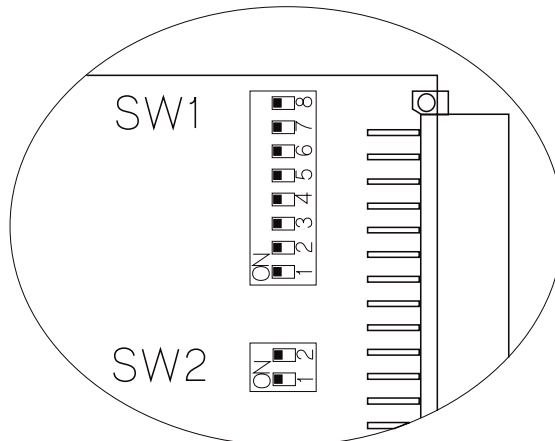
SW1: DIP-switch, 8 -contact, setting of:
- station address
- baudrate

SW2: DIP-switch, 2 -contact, for setting the PROFIBUS line address



3.3.2.2. Setting transfer speed and PROFIBUS station address

There are 2 encoding switches (SW1 and SW2) on the PROFIBUS module:



The following settings of encoding switch SW1 are used for the baudrate and the station address:

Dip switch								PROFIBUS station address
1	2	3	4	5	6			
off	off	off	off	off	off			0
on	off	off	off	off	off			1
off	on	off	off	off	off			2
on	on	off	off	off	off			3
off	off	on	off	off	off			4
on	off	on	off	off	off			5
off	on	on	off	off	off			6
on	on	on	off	off	off			7
etc. through to:								
off	on	on	on	on	on			62
on	on	on	on	on	on			63
1	2	4	8	16	32	←	significance of switch 1-6 (binary coded)	
						7	8	baud rate [kBaud]
								max. line length [m] *)
						off	off	500
						on	off	19.2
						off	on	9.6
						on	on	invalid setting

*) The given line lengths are valid for wire cross sections $\geq 0.22 \text{ mm}^2$. The values in brackets are valid for cross sections $\geq 0.5 \text{ mm}^2$. The lengths are indicated for the whole bus or until the first repeater.



The settings of the PROFIBUS encoding switch are only read during the PC startup sequence. Changes during the operation will not be accepted.

Due to a lack of space, not all settings for station addresses are shown in the table.

Switches 1...6 are binary coded. The station addresses not listed can easily be determined using the significances.

Both switches of the DIP-switch SW2 must always be set to "OFF".

3.3.3. KUAX 680I Profi Control

3.3.3.1. Location of the PROFIBUS interfaces

Both PROFIBUS interfaces of the KUAX 680I are below one another on the left of the front side of the device.

The connection to the PROFIBUS is described on page 3-11.

If, instead of two PROFIBUS connections, the KUAX 680I only has one PROFIBUS connection (RS485) and one V.24 interface (RS232), then the V.24 interface is the upper of the two connections. Use this connector for a programming PC or a user terminal. In this case, the PROFIBUS interface is connected to the **bus** via a T-junction (see page 3-16).



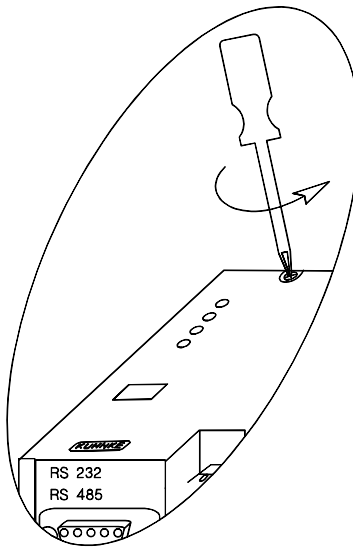
3.3.3.2. Setting transfer speed and PROFIBUS station address

An 8-contact encoding switch (DIP-switch) is located on the power supply board (right at the bottom of the device). It is located underneath the side cover, on which the status light emitting diodes (LED) can be found.

Use this switch to set the PROFIBUS station address and the transfer speed.

Setting the encoding switch:

- loosen the screw with a size 1 cross-recessed screwdriver;
- pull the flap carefully forwards (over the D-sub connectors) and lift it up;
- use a suitable tool (e.g. a thin blade screwdriver) to set the switch according to the table on the next page;
- close the flap again.



Dip switch							PROFIBUS station address		
1	2	3	4	5	6				
off	off	off	off	off	off			0	
on	off	off	off	off	off			1	
off	on	off	off	off	off			2	
on	on	off	off	off	off			3	
off	off	on	off	off	off			4	
on	off	on	off	off	off			5	
off	on	on	off	off	off			6	
on	on	on	off	off	off			7	
etc. through to:									
off	on	on	on	on	on			62	
on	on	on	on	on	on			63	
1	2	4	8	16	32	←	significance of switch 1-6 (binary coded)		
						7	8	baud rate [kBaud]	max. line length [m] *)
						off	off	500	200
						on	off	19.2	1200
						off	on	9.6	1200
						on	on	invalid setting	

*) The given line lengths are valid for wire cross sections $\geq 0.22 \text{ mm}^2$. The values in brackets are valid for cross sections $\geq 0.5 \text{ mm}^2$. The lengths are indicated for the whole **bus** or until the first repeater.



The settings of the PROFIBUS encoding switch are only read during the PC startup sequence. Changes during the operation will not be accepted.

Due to a lack of space, not all settings for station addresses are shown in the table.

Switches 1...6 are binary coded. The station addresses not listed can easily be determined using the significances.

3.3.4. KUAX 680S Profi Control

3.3.4.1. Location of the PROFIBUS interfaces

Both PROFIBUS interfaces of the KUAX 680S are below one another on the left of the front side of the device.

The connection to the PROFIBUS is described on page 3-11.

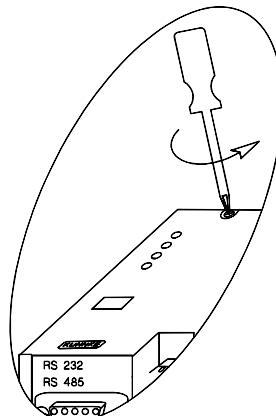
3.3.4.2. Setting the PROFIBUS protocol and the PROFIBUS station address for the KUAX 680S

A 10-contact encoding switch (DIP-switch) is located on the power supply board (right at the bottom of the device). It is located underneath the side cover, on which the status light emitting diodes (LED) can be found.

Use this switch to set the PROFIBUS protocol (PROFIBUS-FMS or PROFIBUS-DP) and the PROFIBUS station address.

Setting the encoding switch:

- loosen the screw with a size 1 cross-recessed screwdriver;
- pull the flap carefully forwards (over the D-sub connectors) and lift it up;
- use a suitable tool (e.g. a thin blade screwdriver) to set the switch according to the table on the next page;
- close the flap again.





Use DIP-switches 1-7 to set the station address according to the table below. For the SINEC L2-DP bus, the **master** is assigned station address no. 2. The **slaves** are assigned station addresses greater 3. Assignment should leave no gaps in the sequence of addresses.

Dip switch							PROFIBUS station address		
1	2	3	4	5	6	7			
off	off	off	off	off	off	off			0
on	off	off	off	off	off	off			1
off	on	off	off	off	off	off			2
on	on	off	off	off	off	off			3
off	off	on	off	off	off	off			4
on	off	on	off	off	off	off			5
off	on	on	off	off	off	off			6
on	on	on	off	off	off	off			7
etc. through to:									
on	off	on	on	on	on	on			125
off	on	on	on	on	on	on			126
1	2	4	8	16	32	64	← significance of switches 1-7 (binary coded)		
							8		bus protocol
							off		PROFIBUS-FMS
							on		Sinec L2-DP
							9		reaction to bus errors
							off		all outputs off
							on		status of outputs unmodified
							10		reaction to undervoltage
							off		all outputs off
							on		outputs can still be changed (no guarantee. We recommend: analyse error 2 in master)

Due to a lack of space, not all settings for station addresses are shown in the table.

Switches 1...7 are binary coded. The station addresses not listed can easily be determined using the significances.

DIP-switch #8 must be set to "ON" to define the KUAX 680S as a DP-slave. Setting DIP-switch #8 to "OFF" selects the protocol PROFIBUS-FMS.

Use DIP-switch #9 to select the reaction of the outputs of the 680S to bus failure:

DIP-switch "ON"	outputs are "frozen" after bus failure
DIP-switch "OFF"	outputs are switched off after bus failure

DIP-switch #10 is used to set the reaction to undervoltage.

3.3.4.3. Transfer speed

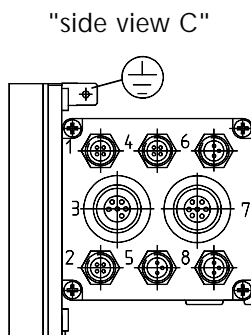
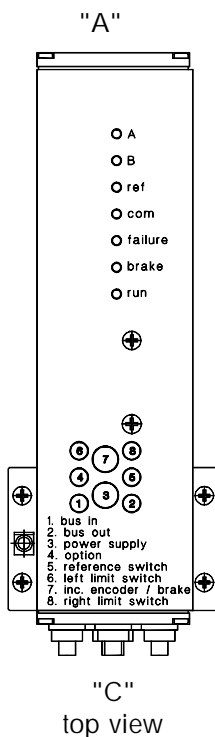


The transfer speed (max. 500 kbaud) need not be set for the KUAX 680S. The device automatically adapts to the speed given by the **master**.

3.3.5. KUAX 681M Profi Control

3.3.5.1. Location of the PROFIBUS interfaces

The PROFIBUS connections of the KUAX 681M are located on side "C" of the top view of the device:



1. male 4-pin connector
- fieldbus IN (RS 485)
2. female 4-contact connector
- fieldbus OUT (RS 485)
3. male 6-pin connector
- power supply for KUAX 681M motor
4. female 4-contact connector
- 1 input IRQ
- 1 optional input (manual on-off push button switch)
5. female 3-contact connector
- reference switch
6. female 3-contact connector
- limit switch left
7. female 7-contact connector
- incremental encoder and brake
8. female 3-contact connector
- limit switch right

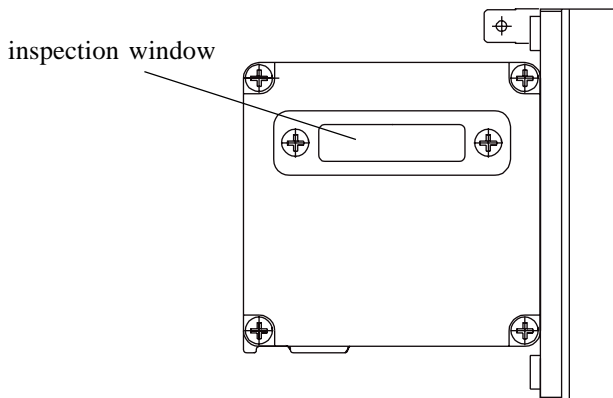
Hardware

These are the 4-contact connectors #1 and #2 which are located right at the bottom of the device. Connector #1 is a male connector and #2 is a female one. Connector #1 is used for the cable from the preceding PROFIBUS station, connector #2 is used for the cable going out to the subsequent bus station.

T-junction and spur line are not required for this device.

3.3.5.2. Setting transfer speed and PROFIBUS station address

There is an 8-contact encoding switch behind the window on the right side of the device (see top view, view A). This is used to set the PROFIBUS station address and the transfer rate. Unscrew the window to access the switch:



Use the table below to set the encoding switches:

Dip switch						Field bus station address	
1	2	3	4	5	6		
off	off	off	off	off	off		impermissible setting
on	off	off	off	off	off		1
off	on	off	off	off	off		2
on	on	off	off	off	off		3
off	off	on	off	off	off		4
on	off	on	off	off	off		5
off	on	on	off	off	off		6
on	on	on	off	off	off		7
etc. through to:							
off	on	on	on	on	on		62
on	on	on	on	on	on		63
1	2	4	8	16	32	←	significance of switch 1-6 (binary coded)
Profi Control: (PROFIBUS)						7	8
						off	off
						on	off
						off	on
						on	on
							impermissible setting

*) The given line lengths are valid for wire cross sections $\geq 0.22 \text{ mm}^2$. The values in brackets are valid for cross sections $\geq 0.5 \text{ mm}^2$. The lengths are indicated for the whole **bus** or until the first repeater.



The settings of the PROFIBUS encoding switch are only read during the PC startup sequence. Changes during the operation will not be accepted.

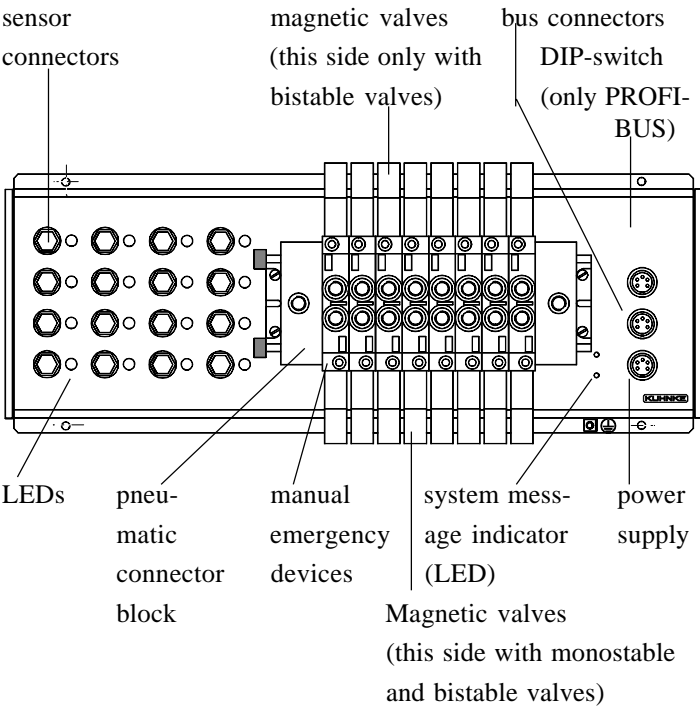
Due to a lack of space, not all settings for station addresses are shown in the table.

Switches 1...6 are binary coded. The station addresses not listed can easily be determined using the significances.

3.3.6. Valve Island 799 PROFIBUS

3.3.6.1. Location of the PROFIBUS interfaces

Both of the RS-485 interfaces of the Valve Island 799 are located on the right and on the top of the device. They are marked with "BUS in" and "BUS out". The cable coming from the previous bus station is connected to the "BUS in" connector, and the cable leading to the following bus station to "BUS out".



3.3.6.2. Setting the PROFIBUS station address and the reaction of the outputs to system failures

The 10-contact encoder switch to set the station address is on the top side of the Valve Island, on the right above the bus connections behind the window. Unscrew the inspection window to access the DIP-switch.

The DIP-switches are set according to the following table:

Dip switch							PROFIBUS station address		
1	2	3	4	5	6	7			
off	off	off	off	off	off	off			0
on	off	off	off	off	off	off			1
off	on	off	off	off	off	off			2
on	on	off	off	off	off	off			3
off	off	on	off	off	off	off			4
on	off	on	off	off	off	off			5
off	on	on	off	off	off	off			6
on	on	on	off	off	off	off			7
etc.									
through to:									
on	off	on	on	on	on	on			125
off	on	on	on	on	on	on			126
1 2 4 8 16 32 64 ← significance of switches 1-7 (binary coded)									
						8			no function
						9			reaction to bus errors
						off			all valves off
						on			status of valves unmodified
						10			reaction to undervoltage in the valve supply
						off			all valves off
						on			status of valves unmodified



The settings of the PROFIBUS encoding switch are only read during the PC startup sequence. Changes during the operation will not be accepted.

Due to a lack of space, not all settings for station addresses are shown in the table.

Hardware

Switches 1...7 are binary coded. The station addresses not listed can easily be determined using the significances.

The reaction of the outputs to bus failures or undervoltage in the supply of the valves is set by DIP-switches 9 and 10.

3.3.6.3. Transfer speed



The transfer speed need not be set for the Valve Island. The device adapts itself to the baudrate given by the **masters**.

3.4. Programming of controllers via different interfaces

The KUAX 644, KUAX 657P and KUAX 680I must be programmed.

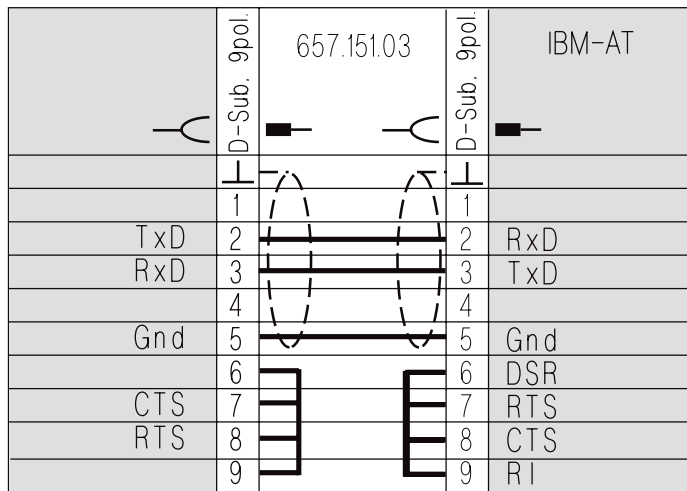
Programming can basically be carried out in 3 different ways:

- via the V.24 interface (all, except KUAX 680I with 2 PROFIBUS interfaces)
- via the PROFIBUS (all)
- via the PC bus (KUAX 644)

The KUAX 680S is a pure input and output device and is not programmed; the same goes for the Valve Island 799.

3.4.1. Programming via the V.24 interface

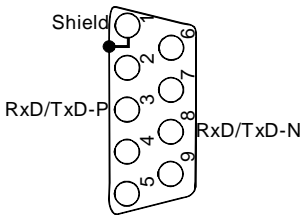
The V.24 interface of the controller (usually the upper plug connector) is connected to a serial interface of the programming PC by a programming cable (see appendix).



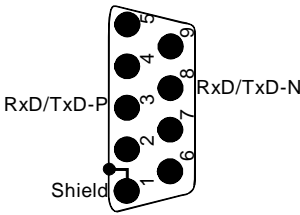
Hardware

Layout of ontacts

female connector



male connector



The program is transmitted from within the KUBES programming software using a fixed **protocol**: 9600 baud, odd parity, 8 data bits, 1 stop bit.

The necessary entries in your WIN.INI were automatically entered when KUBES was being installed on your PC:

```
[ports]
COM1:9600,o,8,1

[KUBES]
SPS=COM1:
```


3.4.2. Programming via the PROFIBUS

The programming PC must be equipped with a PROFIBUS interface before you can program the controller via the PROFIBUS (see page 3-5).

The set PROFIBUS station address of the PROFIBUS module installed must have been told to the PC by the corresponding entry in the file WIN.INI (see page 3-19).

For programming, the programming PC is connected to the PROFIBUS like every other station, i.e. by its PROFIBUS interface. The PC then takes part in PROFIBUS communication as a separate bus station with its own station address.

3.4.3. Programming via the PC bus (KUAX 644)

The fastest connection to programming a KUAX 644 via the PC is provided by the PC bus. In this case, the KUAX 644 is addressed via accesses to the port address area.

Data exchange is carried out via a 1 Kbyte dual-port RAM (DP-RAM). This memory area is addressed by the PC via 16 blocks in the port area of an IBM-AT compatible computer of 64 bytes length each.

There is a total of 4 areas in the port address area of AT computers which can be used as input and output channels.

These are the following areas:

- \$0100 ... \$01EF = 240 byte
- \$0220 ... \$026F = 80 byte
- \$0280 ... \$02AF = 48 byte
- \$0320 ... \$035F = 64 byte

Hardware

Two of these areas of 64 bytes length each are used and each is mapped eight times with 1K offset on the 64K address area. Thus a maximum of 8 KUAX 644 can be used at any one time.

Each of the following addresses represents the beginning of a 64 byte area. All 16 areas together make up the 1 Kbyte DP-RAM.

The following addresses can be used:

Card	Basis	Off 0	Off 1	Off 2	Off 3	Off 4	Off 5	Off 6	Off 7
		Off 8	Off 9	Off A	Off B	Off C	Off D	Off E	Off F
1	\$0000	\$0120	\$0320	\$0520	\$0720	\$0920	\$0B20	\$0D20	\$0F20
		\$1120	\$1320	\$1520	\$1720	\$1920	\$1B20	\$1D20	\$1F20
2	\$2000	\$2120	\$2320	\$2520	\$2720	\$2920	\$2B20	\$2D20	\$2F20
		\$3120	\$3320	\$3520	\$3720	\$3920	\$3B20	\$3D20	\$3F20
3	\$4000	\$4120	\$4320	\$4520	\$4720	\$4920	\$4B20	\$4D20	\$4F20
		\$5120	\$5320	\$5520	\$5720	\$5920	\$5B20	\$5D20	\$5F20
4	\$6000	\$6120	\$6320	\$6520	\$6720	\$6920	\$6B20	\$6D20	\$6F20
		\$7120	\$7320	\$7520	\$7720	\$7920	\$7B20	\$7D20	\$7F20
5	\$8000	\$8120	\$8320	\$8520	\$8720	\$8920	\$8B20	\$8D20	\$8F20
		\$9120	\$9320	\$9520	\$9720	\$9920	\$9B20	\$9D20	\$9F20
6	\$A000	\$A120	\$A320	\$A520	\$A720	\$A920	\$AB20	\$AD20	\$AF20
		\$B120	\$B320	\$B520	\$B720	\$B920	\$BB20	\$BD20	\$BF20
7	\$C000	\$C120	\$C320	\$C520	\$C720	\$C920	\$CB20	\$CD20	\$CF20
		\$D120	\$D320	\$D520	\$D720	\$D920	\$DB20	\$DD20	\$DF20
8	\$F000	\$E120	\$E320	\$E520	\$E720	\$E920	\$EB20	\$ED20	\$EF20
		\$F120	\$F320	\$F520	\$F720	\$F920	\$FB20	\$FD20	\$FF20



If you have installed more than one PC module in your system you must make sure that these do not use the same addresses. If other modules outcode the address area only incompletely, intersections in the area of several basic addresses will occur and the KUAX 644 will not work reliably.

In this case, please remove the corresponding module(s) from the PC and contact the responsible manufacturer.

The address set on the module (KUAX 644) must be transmitted to the KUBES driver. This happens through an entry in your WIN.INI which is not automatically made during installation:

[KUBES]

AT_SPS_a=ST,0xcxxx

This individual parameters have the following meaning:

a serial number of the KUAX 644 (1...8)
 ST PROFIBUS station address of the KUAX 644
 cxxx 120, 2120, 4120, 6120, 8120, A120, C120, E120
 this is the basic address of the module in the PC

Example:

[KUBES]

AT_SPS_1=11,0x120

The corresponding KUAX 644 then has the station address 11 and is addressed by the driver as from port address \$120.



Please make sure that the basic address set on the KUAX 644 board (see instruction manual of the KUAX 644, E331GB, page 3-8) corresponds to the entry made in your WIN.INI.

Hardware

4. Software

4.1. Programs

Kuhnke provide several programs which are of importance in connection with PROFIBUS:

- KUBES is the user software for programming programmable logic controllers (PLCs).
- BIBS is the library management program for updating the KUBES module libraries. It comes to you in one package with KUBES.
- GrafKEd is a graphic editor for the programming of sequential function charts. It is used in connection with KUBES.
- VEBES is the Kuhnke PROFIBUS network configurator. It is used to set the parameters of the stations connected to your PROFIBUS network and also to define the communication relationships among them. VEBES also works together with KUBES.

The chapters 4.2 and 4.3 below explain all major KUBES and VEBES functions, which are important for working with PROFIBUS. For a detailed documentation please refer to the online help systems of the programs.

4.1.1. VEBES and KUBES: main purpose of the programs



Use VEBES to create **Network projects** and to configure the PROFIBUS network.

You need to define the following parameters:

- the PROFIBUS stations. These can be Kuhnke devices or PROFIBUS devices of other manufacturers;
- the **communication relationships** between these stations;
- the **bus parameters**



- the type of communication (**process chart** or **block transfer**) and the communication protocol (**FMS** or **DP**);
- parameters of new types of devices, which are not listed among the participating stations.

Use KUBES to do the following:

- open network projects and write new programs;
- write controller programs;
- test your programs;
- transmit the completed and tested programs into the connected controllers;
- monitor the PROFIBUS network.

Both programs work closely together. They therefore support switching between them.



To avoid network and data file management conflicts we recommend having only one of the two programs active at a time.

All online functions with active controllers are controlled from within KUBES.

4.2. Using VEBES to configure a PROFIBUS network

VEBES is used to configure PROFIBUS networks of Kuhnke devices and/or devices of other manufacturers. Some frequently used devices of other manufacturers are included in the VEBES station selection list. This much facilitates their connection to the network as many parameters have already been defined. It is also possible to work with other devices or newly developed controllers by Kuhnke, however, which are not included in the list of available devices. To do so you use the VEBES "Devices" menu to define the function of these devices and their respective communication requirements (**service access points, object dictionary**). As an example, have a look at the VEBES "Devices" menu which contains definitions of some devices of other manufacturers which have been included in the list already.



The following section first explains how a PROFIBUS network is configured with Kuhnke devices; then comes a network configuration with devices of other manufacturers. For this second section, there is an exercise where you will find clear instructions of how to use VEBES.

4.2.1. Creating a network

Before you can use VEBES to configure a PROFIBUS network you must first create one. The name that you assign to the network in the "Create network" dialog will be used by VEBES for managing the data of all network stations and their communication relationships.

4.2.2. Setting the PROFIBUS parameters



Use item "Bus parameters" of the VEBES "Network" menu to set the time parameters for the **bus** in dependence of the transmission speed.

The time parameters are set in accordance with the baudrate (9.6; 19.2 or 500 kbaud, depending on the length of the PROFIBUS) set on the individual stations.

You can set the following time parameters: **Slot Time** (TSL), **Station Delay Responder** (TSDR), **Target Rotation Time** (TTR) and the **GAP factor** for the bus protocol. You can also choose between the PROFIBUS "FMS" and "DP" **protocols**. Then the time parameters are automatically defined depending on the set baudrate.



*At this point you do **not** define the bus **protocol** but rather when you select the stations. Operating the bus with the DP protocol is only possible if there is only one **master** and otherwise devices exclusively which are marked with "DP" after their name in the list. As soon as a **slave** without this specification, or a second **master** is connected, the bus will run with the FMS protocol automatically.*

You would normally choose the predefined PROFIBUS protocols. These have been optimized for Kuhnke devices.

Alternatively, you have the possibility to define optional time parameters. This is recommended in cases where you have an unusual make-up of stations on the **bus**, e.g. a large number of masters, or if you have a station which cannot work with the predefined parameters.



Clicking on the "free definition" option button activates text fields "**Slot Time**", "**Station Delay Responder**", "**Target Rotation Time**" and "**GAP Factor**". You can now make your own entries. Please consider the necessary/permissible overall cycle time. On the one hand you must not exceed this time; on the other hand, all sensors in the network must be polled at least once, and all actuators must be able to react to changed sensor values in this time.

- "**Slot Time**" is the time between sending out a request and receiving the corresponding response or acknowledgement. Under VEBES, the entry is made as "bit".
- "**Station Delay Responder**" is the response time of the network stations. You are requested to set an upper and a lower limit for this time value. The entry is made as "bit".



The bit time is the time which a bit needs to be transmitted. It entirely depends on the set baudrate.

- "**Target Rotation Time**" is defined as the time interval during which all stations with **bus access right** must have had the token at least once. The time to be set for this parameter depends on the number of active stations (**masters**) on the **bus** and the quantity of data to be transmitted. It must be set to allow each master sufficient time to carry out its message cycles after it has received the **token**. The entry is made as "bit".
- The "**Gap Factor**" is the time in which the network is checked for the possible existence of any new stations. It directly influences the time which a new master needs to be accepted as a new bus station. Under VEBES, this parameter is entered as a multiple of the "**Target Rotation Time**" value.



The **bus parameters** must be the same for all stations. Therefore, you only enter the information once under VEBES; it is then automatically valid for the whole bus. "Target Rotation Time" and "Gap Factor" are redundant parameters for **slaves**.



Wrong entries for the optional time parameters can render bus operation impossible!

4.2.3. Defining bus stations

Use option "Stations" of the VEBES "Network" menu to define the PROFIBUS stations. Defining the stations also determines the PROFIBUS **protocol** (DP or FMS) under which the bus will run. If you select only one master and otherwise only slaves marked with "DP", then the bus automatically uses the SINEC L2-DP protocol. Bus operation is based on the FMS protocol in all other cases.

The following is a list and description of the information entered after choosing the "Station" option:

Symbolic station names. It is practical to enter a name for the station which represents its function. The use of symbolic names aims at making the overview of the network easier.

Station address. It is compulsory to enter the station address which has been or will be set in the device.



*In a PROFIBUS network which you want to operate as a DP bus, the **master** receives station address #2. Assign the station addresses from #3 up to the **slaves**.*

Device type. VEBES has a list of devices from which you can choose the stations. All Kuhnke PROFIBUS machines are included in this list next to a number of frequently used devices of other manufacturers.

If you want to work with devices other than those on the list, you can simply add them to the list. To do so you first define the device type (see option "Type" of the "Device" menu explained on page 4-13).



When setting up PROFIBUS networks for DP protocol operation you must make sure only to use devices marked with "DP" after their name in the list. Also ensure these devices to be set to the DP protocol if necessary (see for example. KUAX 680S, page 3-30).

Configuration. If you choose one of the devices of the list, the default configuration of the corresponding device, i.e. the number of digital and analog inputs and outputs, is automatically taken over. You need only make any entries into the "Configuration" text boxes of those devices which have a variable number of decentralized inputs and outputs (such as the KUAX 680S).



*It is compulsory that the information entered into the "Configuration" text box corresponds to the actual module configuration. It is otherwise impossible to set any outputs when using the FMS **protocol**. When working with the DP protocol the bus does not even initiate the connection when there are differences between the actual and the set number and type of modules.*

4.2.4. Defining connections



You have to define which station you want to communicate with which other station or station (**communication relationship list**).

Two different types of communication are possible under VEBES: communication via the **process chart** or the so-called **block transfer**.

Communication via the process chart

For this type of communication, address ranges are reserved in the memories of the participating **masters**. Communication with the connected stations then runs automatically via these address ranges. VEBES supplements the operand list of the master by the agreed communication objects (external operands, see page 4-28) of the connected stations, and distributes it across the **process status memory**. In Kuhnke devices, the process status memory available for the process chart has a size of 496 bytes. From this memory area, sections (frames) are allocated to the inputs (I) and outputs (O) of each station; the frames can again be sub-divided into sections for analog (A) and digital (D) signals. External operands will then be read as inputs or addressed as outputs via PROFIBUS.

The status of the external operands is written into the process status memory at set time intervals which do not synchronize with the PLC-cycle. Updating can thus also take place during the processing of a module. Interrupt and timer modules cannot be interrupted by updating processes.



As updating can occur while the process has run half way through a module, it can happen that external operands have different values at the beginning and at the end of a module.



If an output of a **slave** (e.g. of a KUAX 680S) is to be activated or deactivated, then the user program in the **master** writes this information into the **process chart**. It is then transmitted straight to the slave via PROFIBUS. This operation, i.e. sending, is not repeated cyclically but only when there are status changes in the process chart.



Unplugging and re-plugging a slave output module during operation leads to a loss of the status information of all outputs of this module. These outputs are only updated again if a status change occurs in the process chart.

The process chart structure of Kuhnke stations with a controller **profile** (KUAX 680I, KUAX 657P, KUAX 644) is permanently set (in a later VEBES version it will be possible to configure it according to user definitions). At present there are 8 bytes available for input signals (groups 00..07) and 8 bytes for output signals (groups 00..07).

The size of the process chart for slaves depends on the configuration. With the KUAX 680S, for example, each connected input and output group is assigned one byte in the process chart. There are modules with one or two groups. 16-contact modules have 2 groups, e.g. 16 inputs or 8 inputs and 8 outputs.

Communication between 2 **masters** takes place between the frames, not directly between the inputs and outputs. It is thus guaranteed that a master has the sole access right to its periphery. The other master has no access to it. Communication between two masters is a write only process, i.e. one master cannot read the memory area of the other master.

For a master to write output information to a **slave** and to read the input information of that slave always takes two program cycles..

Block transfer



Block transfer is used to transmit large data blocks. Transmitting or receiving data blocks is controlled via corresponding commands in the user program. Transmitting, or sending, means to write the data of a data block (own block number) into a block of the other station (partner block number). Receiving means reading data from the block of the other station (partner block number) and writing it into one's own block (own block number). Both data blocks must have the same length. The starting addresses are defined by the user.

Block transfer is possible between all devices, but only with the **FMS protocol** and not the DP protocol.

KUBES modules PB_SEND, PB_REC and PB_STAT take over the handling of the block transfer jobs in the user program of the controllers concerned (see page 4-37).



*In SINEC-L2-DP operation, communication is only possible via the **process chart**.*

In connections between two masters the sending or receiving of a block is triggered in only one of the two user programs. The partner has no program entries to signal the ready-to-receive state, for example. Sending and receiving operations are services unrequested by the partner.

4.2.4.1. Process chart communication connections

For this type of communication connections choose option "Connections - Process chart" of the VEBES "Network" menu.

VEBES lists all defined stations in two lists. Define the connections by first selecting (highlighting) a station in the left list box ("from") and then selecting one in the right list box ("to"). If you have selected two matching stations, then you can enter the connection between the two into another list box ("Existing connections"). When all connections have been defined, VEBES creates a database from the list of connections, the **communication relationship list**, the symbol table for the process chart and the tables of operands.



4.2.4.2. Block transfer communication connections

Choose option "Connections - Block transfer" of the VEBES "Network" menu. VEBES displays two lists of defined stations. From the left list box ("Transfer block from"), select the station from which you want to transmit a block, and from the right list box ("Transfer block to"), select the receiving station.



*These lists only include Kuhnke devices for which you have defined **block transfer** via option "Block transfer objects" in the "Devices" menu.*

For handling block transfer objects of **masters** you must enter the **OD indices** (block numbers). Block numbers 160-169 directly correspond to OD indices 160-169. The block transfer objects are included in the **object dictionary** already, but "conf." – for "dependent on configuration" – is the default setting for the length of the block. You therefore have to input the starting address (e.g. BM00.00) and the length of the block (e.g. 16 operands) for these operand ranges.

You can make your entries after having clicked on one of the "new" control buttons underneath the list boxes. VEBES will display a list of text boxes into which you can write your information. The "OD index" and "block length" fields expect you to make an entry. The starting address of the block can alternatively be entered under "operand" or under "memory address" and "bank". These inputs can be changed with the help of the control button "change", as long as no connection has been defined.



For block transfer communication objects of **slaves**, the available block numbers (OD indices) for block transfer and the lengths of the blocks are predefined and cannot be changed (see "Devices" menu, option "Block transfer objects", page 4-18). However, clicking on the "Change" button will display the information. The "Connection from" list box only shows block numbers which allow read-only, or read/write access (inputs), while the "Connection to" list box only shows block numbers which allow write-only or write/read access (outputs).

The entered **OD indices** are shown as a list for each active station. Here too you have to activate a block number. When the correct connection has been selected (i.e. a station and the corresponding block number in either of the lists), it will be taken over and saved by VEBES after clicking on the "Accept" button. You can define up to 10 data blocks per station (only valid for Kuhnke masters, block numbers 160..169).



*The data blocks defined for **block transfer** must have the same length in both stations. If they have not, VEBES will display an error message.*

4.2.5. Using VEBES to integrate non-Kuhnke devices into the PROFIBUS network



The procedure is basically the same as the one described above:

- Open/create a network.
- Define or take over the **bus parameters**.
- Define the stations; this also determines the PROFIBUS **protocol** (DP or FMS) to be used.
- Define the connections.

Defining devices by other manufacturers as PROFIBUS stations which are listed in the VEBES station selection list ("Network" menu, option "Stations") is done the same as defining Kuhnke devices.

For other devices, you must first define the device type under VEBES.

The following information is required for defining a device type:

- device name
- selection **master** or **slave**
- creation of the **object dictionary**
- definition of the **service access points**
- definition of the **process chart** objects
- definition of the **block transfer** objects

4.2.5.1. Defining the device type

To define the device type, you first choose option "Type" from the VEBES "Devices" menu. Here you can enter the device name and take it over into the station selection list (see page 4-6).



It is not possible to define SINEC L2-DP devices or to view the definition of these devices. The commands of the "Devices" menu stay dimmed (i.e. they are not available in the present situation).

Selecting either the "Master" or the "Slave" radio button determines the function of the new device in the network.



*VEBES only accepts those devices as **masters** which can be programmed under KUBES, i.e. devices by other manufacturers are automatically treated as **slaves**. Although you can use VEBES to define **objects** and set SAPs for these masters, you will have to import the corresponding information into the programming software first.*

You also determine whether a device may have a variable I/O configuration ("length of process chart objects depends on device configuration"). This piece of information is important, because you have to enter the exact number of inputs and outputs for determining the process chart for the future communication partner when defining the stations (see page 4-6). After you have completed the information required for the device type, the remaining commands of the "Device" menu are activated (i.e. they can now be chosen).

4.2.5.1.1. Object dictionary

The PROFIBUS communication objects for new devices must be entered into the **object dictionary** (VEBES "Devices" menu, option "Object dictionary"). Only the **objects** included in the object dictionary can later be used for **process chart** or **block transfer** communication operations.

An **OD (object dictionary) index** must be entered for each object. Enter a number between 1 and 65535. The OD index is the logical address of the object. OD indices 160..169 are reserved for the block transfer of Kuhnke **masters**.



Additionally, each object must be allocated a **data type**. Under VEBES, the following data types are allowed:

- integer 8, integer 16, integer 32;
- unsigned 8, unsigned 16, unsigned 32;
- octet string.

Octet strings are designed for digital and analog inputs and outputs of Kuhnke devices. For non-Kuhnke devices, please follow the instructions of the manufacturer.

The next step is to set the **object code (object type)**. VEBES provides the following object types:

- simple variable
- array
- event

You can enter a length for the **object**, depending on the data and object types, or you can take over the default value. The length of **process chart** objects can be determined when entering the information for devices whose number of inputs and outputs depends on the configuration (see "Network" menu, option "Station", page 4-7). In this case the "length" to be entered into the **object dictionary** is "conf."

Finally, you decide which access type you want the object to support. The choice is "read-only", "write-only" and "read/write". We recommend using access type "read/write" for **OV indices 160..169 of masters**. These are reserved for **block transfer** communication.

After you have defined all object attributes, they will be taken over into the object dictionary by clicking on "Accept". Upon quitting the "Object dictionary" dialog, VEBES takes over all data entered into the object dictionary and creates a database record.

4.2.5.1.2. Service Access Points



Service access points (VEBES "Devices" menu, option "Service access points (SAP)") must be arranged for new bus stations. Process chart and block transfer communication data can only be transmitted through the SAPs defined in the list of the device.

Copy the data which VEBES requires you to enter into the "Define service access point" dialog from the data sheet of the device to be defined.

Enter a number between 0 and 63 into the "SAP" text box. The NIL-SAP (for frames without the a SAP number) has the number 128.

Defining the type of connection also sets the type of communication partners and the type of communication. There are three **connection types** to choose from under VEBES:

- MMAC, master-master connection supporting acyclic **data transfer services**;
- MSAC, master-slave connection supporting acyclic data transfer services;
- MSAC_SI, master-slave connection supporting acyclic data transfer services and slave initiative.

VEBES does not support cyclic data transfer services.

Apart from defining the type of connection you also have to assign a **connection attribute**. It indicates whether the connection is a defined connection (D), an open connection of the responder (O) or an open connection of the initiator (I).



Into the "Max PDU" field enter the maximum lengths of **protocol data units** (PDUs) for messages with high and low priority in bytes for sending and receiving operations. VEBES automatically suggests default values which depend on the connection type selected. Accept these suggestions or overwrite them.

All other **services** supported by the device are set under "**FMS features supported**". The program expects entries as hexadecimal byte inputs. Here too, VEBES automatically suggests default values which depend on the connection type selected. Accept these suggestions or overwrite them.

Max. SCC, RCC, SAC and RAC are counters for the maximum number of parallel services. **CI** is the cyclic control interval and is normally set to 0.

4.2.5.1.3. Process chart objects

Use option "Process chart objects" of the VEBES "Devices" menu to allocate an **OD index** as defined in the corresponding **object dictionary** to the four different **process chart** objects (I/O binary; I/O analog). Also define a **service access point** which is used jointly by all of the four objects. The allocation is done by means of a list boxes in which VEBES shows the previously defined OD indices and service access points. If you want no OD index to be entered for the objects, then the entry "empty" must be chosen from the OD lists. This is necessary in cases where no analog inputs and outputs are available.



*All defined process chart objects can be addressed via the process chart inside the controller. Communication between **masters** is handled automatically by the masters.*

4.2.5.1.4. Block transfer objects



Use option "Block transfer objects" of the VEBES "Devices" menu to allocate **block transfer** objects to service access points. VEBES displays list boxes containing the previously defined SAPs and OD indices. From these lists, choose a combination of a SAP and an OD index to be added to the list of block transfer objects (object dictionary list). For Kuhnke masters, you can connect up to 10 objects to a service access point. OD indices 160..169 are reserved for block transfer communication.



Communication of block transfer objects must be initiated via KUBES modules in the user program.

Exercise: Defining devices

This example aims at explaining how you define a new device by another manufacturer for the network. The exercise only use the VEBES "Device" menu .



- From the "Device" menu, choose option "Type". This displays the "Define device type" dialog.



- Into text box "Device type", enter "practice" and click on "Accept".

Option buttons "master" and "slave" are dimmed. "Slave" is checked. In our example we are working with a decentralized input/output device. Option button "modify database entry" is also selected. This means that you can enter information into or edit the information contained in the dialogs of the remaining "Device" menu options.

- Checkmark control box "length of process chart objects depends on configuration", as our device will be equipped with modules.



- Click on "OK". VEBES takes over the entries and enables the other options of the "Device" menu.

- Now choose option "Object dictionary" from the "Device" menu. This opens the "Object dictionary" dialog.



- Make the following entries:

OD index	data type	object code	length	access
40	unsigned 16	simple variable	2	r & w
41	octet string	simple variable	conf.	read
42	octet string	simple variable	conf.	r & w
43	octet string	simple variable	2	read

- Click on "Accept" after each line.

OD index	Data type	Object code	Length (Byte)	Access
43	Octet-String	Simple-Variable	2	Read
40	Unsigned16	Simple-Variable	2	R & W
41	Octet-String	Simple-Variable	conf.	Read
42	Octet-String	Simple-Variable	conf.	R & W
43	Octet-String	Simple-Variable	2	Read

Buttons: Help = F1, Cancel, Accept, Delete, Done

- When all entries are complete, quit the dialog box by clicking on "exit".
- Choose option "Service access points (SAP)". This opens the "Service access points" dialog.
- Make the following entries:

Define service access points [SAP]

Station : practice

loc. SAP	Connection type	attribute	max PDU				FMS features supported (Bytes in hex)						max				CI
			send		receive		1	2	3	4	5	6	SCC	RCC	SAC	RAC	
			high	low	high	low											
5	MSAZ_SI	D	0	49	241	241	00	30	00	40	00	50		0			
4	MMAZ	D	0	128	0	241	00	30	00	00	00	00	1	0	0	0	0
5	MSAZ_SI	D	0	49	241	241	00	30	00	40	00	50	1	0	0	1	0

Help = F1 Cancel Accept Delete Done



- When all entries are complete, quit the dialog by clicking on "exit".
- Choose option "Process chart objects".



- Make the following entries:

Define process chart objects

Station : practice

Objects in process chart

SAP : 5

OO index

binary I : 41

binary O : 42

analog I : empty

analog O : empty

Help = F1 Cancel Done



- When all entries are complete, quit the dialog by clicking on "exit".



- Choose option "Block transfer objects". This opens the "Define block transfer objects" dialog.
- Make the following entries:

Define block transfer objects

Station : practice

SAP	OD index	OD list
5	43	40
		41
		42
		43

Buttons: Help = F1, Cancel, Done, Delete

- Quit the dialog by clicking on "exit".

The information about the new device is now complete and has been taken over by VEBES. The information basically corresponds to that of the KUAX 680S. You can now integrate the "practice" device into a PROFIBUS network instead of a KUAX 680S. If you have a KUAX 680S available and if you would like to try out the newly defined device in the PROFIBUS, you can turn to the exercises on pages 4-31 and 4-41.



You have learnt

- how to add new devices to the list of devices
- how to activate the options of the "Device" menu
- how to define an object dictionary and service access points
- how to define objects for process chart and block transfer communication

Software

4.3. Programming network projects with KUBES

After you have configured the PROFIBUS network under VEBES, go "to KUBES" "with project" or "without project". Choose option "with project" if you would like to write the user program for one of the other programmable stations. In this case, a dialog box appears in which you can select the station to be programmed. VEBES creates a project for each station that can be programmed under KUBES, which is shown in the KUBES status line when a station has been chosen. You can process and save these projects under KUBES just like normal single projects that you work with under KUBES.



Network projects belonging to the networks created and administered under VEBES are also directly accessible from within KUBES. Networks can be selected in the dialog box that opens after you have chosen option "Open project" from the KUBES "Project" menu. The networks are listed in this dialog box. Selecting one of the listed network projects opens another dialog box listing the projects belonging to the network (the programmable stations). When a project choice is confirmed, the project is opened under KUBES and can be edited.

VEBES has then already extended the list of operands by the defined communication **objects** of the connected **process chart** stations in these projects. In the user program, they can be addressed as "external operands" just the same as the internal operands of a controller.

The programming of network project stations under KUBES is basically the same as the programming of individual controllers. The programs are structured as modules and these are written and edited in the KUBES Module Editor. You can also run the numerous KUBES test functions over them to test their integrity and proper functioning. Like in single projects, the command sets and the available operands are dependent on the modules and controllers used, here too.

4.3.1. The Symbol Table

In network projects, the overview of the Symbol Table ranges selection list (KUBES "Edit" menu, option "Symbol Table", Symbol Table "Selection" menu, option "Select") is extended by two additional points:

- PROFIBUS messages
- PROFIBUS

The PROFIBUS messages comprise the following operand ranges:

- **PFaxx.yy** error and failure messages
- **PSaxx.yy** status messages
- **PEaxx.yy** event notifications

During PROFIBUS operation, corresponding messages in the form of error codes are read into these operand ranges. They can be requested from there, for example via the KUBES "Display address range".

Selecting PROFIBUS from the overview list displays a list with the PROFIBUS stations with which there is a process chart connection. Choose a station from this list to have its operands available through PROFIBUS appear in the "operand range" list box. This list thus represents the external operands which can be addressed via the PROFIBUS.



If there are only block transfer connections to other stations, no addresses are displayed in this list.

However, the PROFIBUS messages are valid for both forms of communication.

Like local operands, the operands listed under "PROFIBUS" and "PROFIBUS error messages", too, can be provided with symbolic names (mnemonics) and comments.

4.3.2. Addressing external operands in process chart communication



External operands are the **communication objects** which are agreed upon by a master and another station that the master communicates with. Contrary to "local" operands, "external" operands are not directly available in the controller, but communication is done via the process chart instead (see page 4-8).

The **object dictionaries** of the devices of the stations on the station selection list are automatically created under VEBES so that you need not look into it (see page 4-6). You only have to set objects for newly defined devices not contained in the station selection list (see page 4-14 "object dictionary" and **process chart objects** on page 4-17).

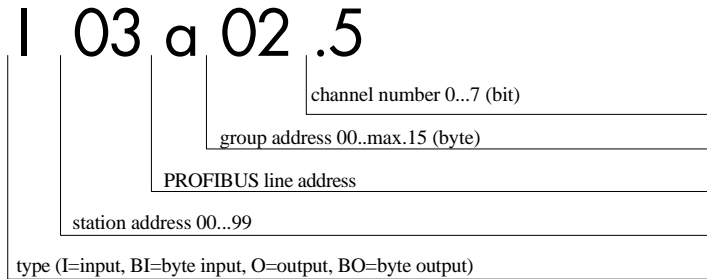
External operands are read as inputs or are addressed as outputs via the PROFIBUS. There are different sources or destinations:

- decentralized input/output devices (e.g. KUAX 680S)
- other programmable logic controllers (e.g. KUAX 680I or KUAX 657P)
- valve islands
- positioning controllers etc.

As the length of the process chart in Kuhnke devices is defined by the controller **profile** class, while the process chart of devices with a sensor/actuator profile class depends on the configuration, there are different aspects to observe for addressing processes. These differences will be discussed in the following. We used a KUAX 680S as an example of a typical slave.

4.3.2.1. Coding of operands

The coded description of external operands is structured as follows:



Type: the relationship between the operand and the controller itself is said through the type.

- Inputs are read via the PROFIBUS (I = bit input, BI = byte input)
- Outputs are set via the PROFIBUS (O = bit output, BO = byte output)

Station address: address of the communication partner on the PROFIBUS

PROFIBUS line address: always "a"

Group address: 8 inputs or 8 outputs form a group. They are numbered from left to right together with the modules connected. If a module has 16 inputs then it has 2 groups. An input/output module has 1 input group and 1 output group (see instruction manual E 307 D, KUAX 680S).

Channel number: Only for bit addressing. This selects a bit from the given group. In a KUAX 680S, the channel number corresponds to the module number, i.e. the channel number represents the input or the output.

4.3.2.2. Addressing

The operands in the process chart can be addressed by bit, byte or word. The same memory is used for either of the different possibilities..

Addressing by bit

This is only possible in the first 4 bytes (groups 00 to 03) of the controllers. Bit addressing serves transmitting binary signals of inputs and outputs or markers. You can address individual channels of a KUAX 680S by bit addressing.

The operand is addressed with "I" (input) or "O" (output). The desired bit is selected by the channel number.

Examples:

```
L  I03a02.5    ; read external input
=  M01.02      ; and write into local marker

L  M00.03      ; read local marker
=  O02a01.7    ; and write into external output
```

Addressing by byte

Addresses 1 byte (8 bit) with one command. All channels of a module of a KUAX 680S are addressed. Inputs are labelled with "BI", and outputs with "BO". You need input no channel number.

Examples:

```
L  BI03a02.    ; read external byte input
=  BM01.00     ; and write into byte marker

L  BM00.00     ; read byte marker
=  BO03a04.    ; and write into the external byte output
```

Addressing by word

Address 1 word (2 bytes or 16 bit) with one command. All channels of 2 groups of a KUAX 680S are addressed. Word addressing in controllers is possible with even group numbers. Inputs are marked with "BI" and outputs with "BO". You need not input the channel number.

Examples:

```
LD    BI03a02.    ; read 2 external byte inputs
=D    BM01.00      ; and write into 2 byte marker

LD    BM00.02      ; read 2 byte marker
=D    BO03a02.     ; write into 2 external byte outputs
```



If external operands are linked in the program (e.g. inputs of a KUAX 680S) then you must make sure that they are valid operands. If, for example, the bus connection is interrupted, then the received value in the process chart may no longer correspond to the actual status of the input. In order to avoid wrong connections resulting from this, we strongly recommend to carefully analyse the PROFIBUS messages in the userprogram (see page 4-62).

On the following pages we have included an exercise to illustrate the setting up of a network and the addressing of external operands.

Exercise: "Addressing operands in the Process Chart"

For this easy exercise, you should first establish a network consisting of a master (e.g. a KUAX 680I, KUAX 644 or KUAX 657P) and a slave (KUAX 680S) (VEBES "Network" menu, option "Open"). Then define these stations (VEBES "Network" menu, option "Station"). Assign station address 2 to the master and station address 3 to the slave.



The station addresses must have been set on the device by the DIP switch.

Define the process chart connections (VEBES "Network" menu, option "Connections - Process chart") between master and slave.



- Change to KUBES "with project".
- Call up the Symbol Table Editor.



- Enter the following into range "Inputs: I00.00-I00.15":

Address	Symbol	Comment
I00.00	IN_LO	local input of the master

- Enter the following into range "Outputs: O00.00-O00.15":

Address	Symbol	Comment
O00.00	OUT_LO	local output of the master

- Choose the option "PROFIBUS" of the "overview" column of the range selection dialog of the Symbol Table Editor. The "station" column lists the stations connected to the PROFIBUS, in this case it is the slave. It has already been activated (highlighted) and its operands are shown in the "operands" column (these are external operands from the master's point of view).



- Choose range I03a00.0-I03a00.7.



- Enter the following:

Address	Symbol	Comment
I03a00.0	IN_EX	slave input

- Now choose range O03a00.0-O03a00.7.

- Enter:

Address	Symbol	Comment
O03a00.0	OUT_EX	slave output

- Close the Symbol Table Editor and change to the Module Editor.
- Load the organization module into the editor and write the following program:

```

MAIN      NOP
; Read local input and write into external output
      L      IN_LO      I00.00      ; (local input of the master)
      =      OUT_EX      003a00.0    ; (slave output)

; Read external input and write into local output
      L      IN_EX      I03a00.0    ; (slave input)
      =      OUT_LO      000.00      ; (local output of the master)
END      NOP

```

This example only uses commands for reading and writing external inputs and outputs. It is obvious that the master treats the external operands like local operands.

- If you like, you can transfer the program into the controller and try it out.

The program contains no elements which serve to control PROFIBUS. However, it is possible to monitor the bus operation through PROFIBUS (see chapter 4.3.5. "Using KUBES to monitor the PROFIBUS network").

- Start the program in the master and call up the "Display single address".
- Enter addresses PFa00.00, PSa00.03 and PEa00.03.

The operating state of the ALI (application layer interface) is shown in operand PFa00.00. Operation is normal (no error detected), if this operand is "8". The partner status of the partner with the station address 3 (the slave) is shown in operand PSa00.03. The zero indicates "normal operation". The error messages of the slave (station address 3) are shown in operand PEa00.03. "Zero" means "no error messages". See page 4-62 for a detailed list of the PROFIBUS messages.



The PROFIBUS can also be monitored via the "Display address range" option. In the range selection dialog, operand ranges PEa, PFa and PSa are listed under "PROFIBUS messages".



When the system is connected, you should run the PROFIBUS only with programs providing a bus monitor and suitable measures in the case of errors or failures. The exercises at the end of the software section are concerned with this type of bus monitoring.



You have learnt that:

- external operands are treated like local operands, but possess their own syntax;
- external operands are available in the overview of the symbol table and the "Display address range" window;
- operand ranges PFa, PSa and PEa serve monitoring the bus.

4.3.3. Programming block transfer (only PROFIBUS-FMS protocol)



A Kuhnke **master** can have **block transfer connections** to 10 different partners. The connection is made via **SAPs**. Blocks 160-169 are reserved for block transfer. A maximum of 230 bytes of user data can be transferred per block.

The object of the communication partner must be of the same length (number of octets) as the block in the master.

A block in a master can communicate with different objects of the partner through its SAP (one after another), as long as the objects are of the same length. The data must be correspondingly copied in the master.

The master can process 8 block transfers at any one time. The **ALI** provides no more capacity.



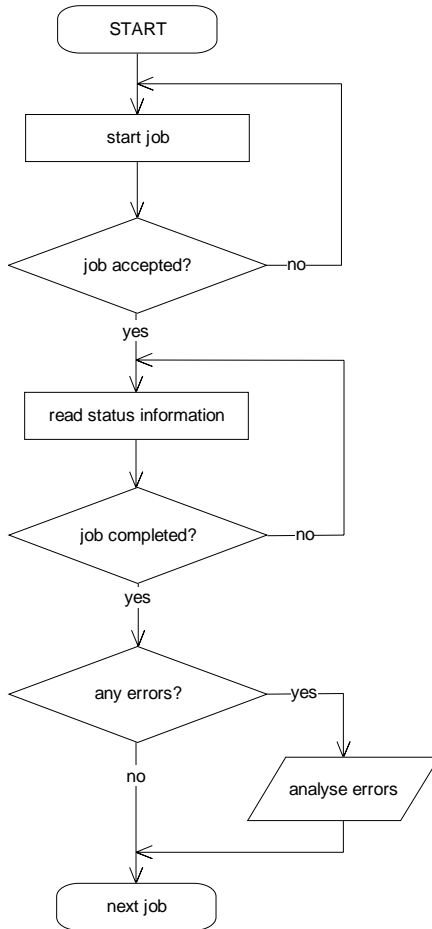
Always analyse the status messages of the KUBES modules carefully.

Before starting the programming, the partners for the block transfer (VEBES "Network" menu, option "Station", see page 4-6) and the data blocks necessary for block transfer (VEBES "Network" menu, option "Connections - Block transfer") should have already been defined.

Communication by block transfer is handled by calling up KUBES modules in the user program. The programming of the block transfer is reproduced in the following schematic program diagram.



If the PLC is reset by KUBES (command "Stop and Reset"), then decentral outputs of a slave (e.g. 680S) which were set by the block transfer will not be reset. Reason: the command "Stop and Reset" is not recognized by the slave.

Schematic program diagram of a block transfer

The flowchart only describes the treatment of the sequences required for block transfer. Wait loops must not be transferred and realised in the user program because this would interrupt the PLC cycle.



Block transfer between master and slave

For block transfer communication between a master and a **slave**, you must always take into account that a slave can only process one **block transfer** operation at a time. The program in the **master** must consider this by sending out the next job for the same slave only when the previous job has been completed. Otherwise, error messages must be anticipated which unnecessarily hold up the program run.

Block transfer between master and master

In master-master **connections**, sending or receiving operations via block transfer are only initiated in one of the user programs. In the program of the partner, **no** ready-to-receive or ready-to-send status signal will be given. The partner will not request sending or receiving operations.

Block transfer communication takes place acyclically to the PLC program. The PLC works as a multi-tasking system. Each active task only has a certain amount of time available, then it will be interrupted and the next task will be called up (pre-emptive multi-tasking). Typical tasks are:

- PLC program
- programmable timers
- PROFIBUS (ALI)
- V.24 communication (e.g. programming PC or user terminal)



Because of this interruption, it can happen, for example, that a block is only partly in its target area while the active master has already received a positive acknowledge signal from PROFIBUS.

Acknowledged block transfer

To receive a "genuine" partner response during block transfer communication between 2 masters, we recommend using the "acknowledged block transfer" method. Please refer to network "PB_BSP_7" on the enclosed exercise diskette for an example program of how this type of block transfer communication can be realised.



A block is always transferred in the order lowbyte to highbyte. In acknowledged block transfer operation, the last byte of a block is thus used as the acknowledgement signal.

Block transfer communication with several partners

A master can process several tasks at a time (max. 8) and can enter parallel communication with various partners. In this case the program sequence is the same for the individual partners although with different parameters.

4.3.3.1. KUBES modules for block transfer

KUBES module PB_SEND

Task to send a data block to the communication partner.

KUBES module PB_REC

Task to receive a data block from a communication partner.

KUBES module PB_STAT

Status request of a task.

4.3.3.2. Parameters of the KUBES modules

Byte markers are used to transfer the input parameters of the KUBES modules. This means that the desired values must be written into the corresponding byte markers before calling up the KUBES module.

<Partner Station Address>, size: 1 byte

Station address of the communication partner for which the data are destined, or from which they are requested.

<Own Block Number>, size: 2 bytes

Number of the data block which is on one's own station.

The data which are in this memory area are transferred to the partner in a sending task (module PB_SEND).

The data received from the partner are written into this memory area in a receiving task (module PB_REC).

<Partner Block No.>, size: 2 bytes

Number of the data block on the communication partner.

The data sent during a sending task (module PB_SEND) are written into this memory area.

The data will be read from this memory area in a receiving task (module PB_REC).

<Task Number>, size: 1 byte

After a task has been placed, the KUBES module (PB_SEND or PB_REC) writes an identifying code number into parameter <task number> which is necessary for the further processing of the task. This is the number that module PB_STAT reads when requesting the status of a task.

If several tasks for sending and receiving data are initiated at the same time, then it is compulsory to use a separate byte for each task.

<Status>, size: 1 byte

Into this byte, the KUBES modules write a <status> signifier as information about the current status of the specific task. The status messages can be shown in the "Display address range", the "Display single address" or the "Dynamic" display in the Module Editor.

Status messages of KUBES Module PB_SEND

Signifier	Description
40	send job accepted -> poll status (PB_STAT)
41	job queue overflow, send job rejected -> must be repeated

Status messages of KUBES module PB_REC

Signifier	Description
40	receive job accepted -> poll status (PB_STAT)
41	job queue overflow, receive job rejected -> must be repeated

Status messages of KUBES module PB_STAT

Signifier	Description	
7	after send job	send job successfully completed *)
8		conflict of types
9		data range does not exist
10		partner error
11		own data range not free
12		partner data range not free
13		connection error
14	after receive job	receive job successfully completed *)
15		conflict of types
16		data range does not exist
17		partner error
18		own data range not free
19		partner data range not free
20		connection error
40	general messages	send or receive job in process
42		invalid job number entered

*) Messages "7" and "14" only indicate that PROFIBUS has completed the job. This does not guarantee, however, that ALI has finished transferring all data into the destination data range (see page 4-37, Acknowledged block transfer)

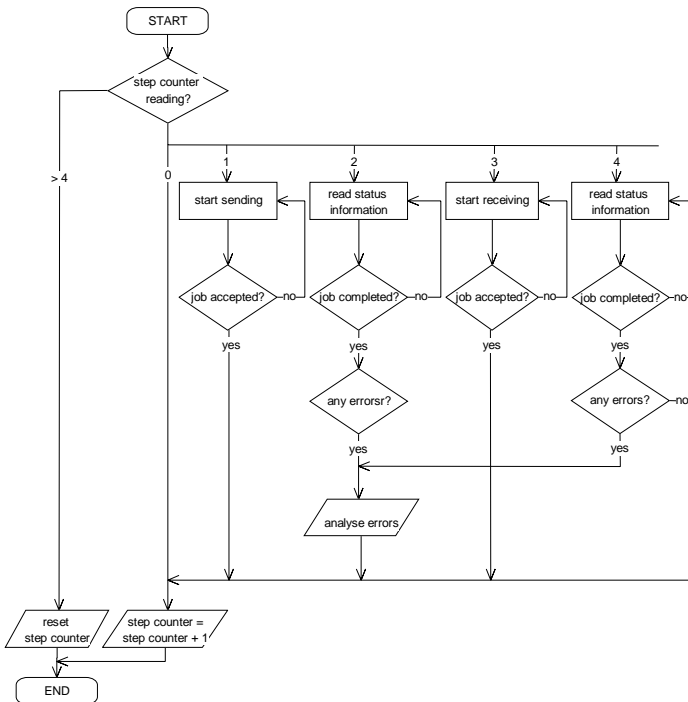
We have included two exercises to illustrate the programming of a block transfer. The first of these exercises is concerned with the programming of a cyclic block transfer communication and the other one with block transfer on command. The actual program contains explanations in the form of comments. You can take these over into your own program as memory joggers or you can leave them out. These and other examples can be found on the example diskette enclosed in this instruction manual.

Exercise: "Cyclic block transfer" (single master network)

The purpose of this example is to explain the process of block transfers. The structure of the program is presented in the flow chart below. A master (stat.1) communicates with a slave through block transfer. Sending and receiving is called up cyclically.



The whole example can be found on the example diskette. The network has the name "PB_BSP_3".



The KUAX 680S is equipped with 8 inputs and 8 outputs.



- The first step is to create a new network under VEBES and to take over the bus parameters of the FMS protocol.

- Define the following stations:

master 01 stat. address 1 device type KUAX 680I

slave02 stat. address 2 device type KUAX 680S

- Choose the option "Connections - block transfer" of the "Network" menu. The dialog box "Define block transfer connections" will appear.

- Click on the master in the "source" box.

- Click on the corresponding "new" button. This opens the "Define block" dialog box.



- Make the following entries:

OD index: 160

Operand: SBM00.00

Length of block (byte): 1

- Click on "OK". The dialog box closes.

- Click on the slave in the "destination" box.

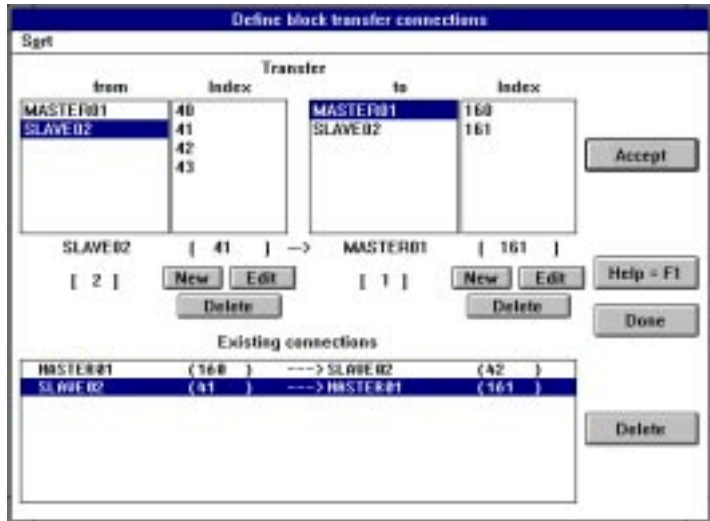
- In the "OD index" box, click on "42".

- Click on "Accept". The defined connection appears below in the list "Existing connections".

- Use the following information to correspondingly define the connection from slave to master:

OD index slave: 41 **operand:** SBM01.00

OD index master: 161 **length of block:** 1



- Close the dialog box by clicking on "Exit".

- Change to KUBES "with project".



- Open the Symbol Table Editor and enter the following:

Address	Symbol	Comment
BM00.00	PAR_NR_2	PB (block trans) partner number
BM00.01	OWN_BL_2	PB (block trans) own block number
BM00.03	PAR_BL_2	PB (block trans) partner block number
BM00.05	ORD_NR_2	task number
BM00.06	PB_STAT_2	task status
BM00.14	ST_CNT_2	step counter (block trans) stn.02
BM00.15	ER_CNT_2	error counter (block trans)stn.02
SBM00.00	S_BLC_02	block to be sent to stn. 02
SBM01.00	R_BLC_02	block to be received from stn.02



- Quit the Symbol Table Editor.

- Open the Module Editor and load the organization module into the editor.

The main program controls the actual process and starts communication via block transfer.

- Write the following program:

```
; ***** Main program *****
MAIN      NOP
; processing block transfer
          JPP      BLCTRA02          5
; writing 8 local inputs into 8 external outputs
          C1T8     I00.00
          =        S_BLC_02          SBM00.00 ; (block to be sent to stn. 02)
; writing 8 external inputs into 8 local outputs
          L         R_BLC_02          SBM00.01 ; (block to be received from stn. 02)
          C8T1     O00.00
; *****
```

- Save the module and load program module "BLCTRA02" into the editor.

In our example we want to send a block to the 680S and also receive one block from the 680S.

Sending:

Master index 160 (SBM00.00) -> 680S index 42 (8 outputs)

Receiving:

Master index 161 (SBM01.00) <- 680S index 41 (8 inputs)

Sending and receiving are to be initiated cyclically.

- Write the following program:

Exercise: Cyclic block transfer

```
; ***** Block transfer to/from Station 02 *****
; Step control
; Communication problems with partner?
; -----
;           This request is only valid for controllers with monitor versions
;           smaller 4.20.
;           As from version 4.20 the next 3 program lines must be re-
;           moved.
; -----
;           L      C_STAT02      PSa00.02 ; (communication status station 2)
;           JP<>   CLR_STEP      ; error -> starting position
;           NOP                      ; no error -> continue
; -----
; Branch strip to the steps
; -----
STEPS      L      ST_CNT_2      BM00.14 ; (step counter (block trans.) stn. 02)
;           CMP      0
;           JP=     NXT_STEP      ; -> next step
;           CMP      1
;           JP=     SND_ORD      ; -> process send job
;           CMP      2
;           JP=     SND_STA      ; -> status request send job
;           CMP      3
;           JP=     REC_ORD      ; -> process receive job
;           CMP      4
;           JP=     REC_STA      ; -> status request receive job
;           JP      CLR_STEP      ; -> reset step chain
; -----
; Set next step
; -----
NXT_STEP   L      ST_CNT_2      BM00.14 ; (step counter (block trans) stn. 02)
;           CMP      4                      ; arrived at last step?
;           JP>=    CLR_STEP      ; yes -> clear step counter
;           INC     ST_CNT_2      BM00.14 ; (step counter (block trans) stn. 02)
;           JP      END_STEP
```

Software

```

; -----
; Reset step chain
; -----
CLR_STEP CLR      ST_CNT_2          BM00.14 ; (step counter (block trans) stn. 02)

END_STEP JP      END

; ***** Sending a Block *****
; -----
; Initiate send job
; -----
SND_ORD  L        2                  ; partner number
        =        PAR_NR_2          BM00.00 ; (PB (block trans) partner number)
        LD        160
        =D        OWN_BL_2          BM00.01 ; (PB (block trans) own block number)
        LD        42
        =D        PAR_BL_2          BM00.03 ; (PB (block trans) partner block no.)
        JPK       PB_SEND , _____
                PAR_NR_2 -| _____| - ORD_NR_2,
                OWN_BL_2 -| _____| - PB_STA_2,
                PAR_BL_2 -| _____| -
        L        PB_STA_2          BM00.06 ; (PB (block trans) job status)
        CMP       41                  ; job queue overflow?
        JP=       END                ; yes -> repeat in next cycle
        JP        NXT_STEP           ; no -> status request

; -----
; Status request of send job
; -----
SND_STA  JPK       PB_STAT , _____
                ORD_NR_2 -| _____| - PB_STA_2
        L        PB_STA_2          BM00.06 ; (PB (block trans) job status)
        CMP       40                  ; send job still being processed?
        JP=       END                ; yes -> check again in next cycle
        CMP       7                   ; send job completed?
        JP=       NXT_STEP           ; yes -> next job
        JP        S_ERROR            ; else -> error analysis

```

Exercise: Cyclic block transfer

```

; ***** Receiving a Block *****
; -----
; Initiate receive job
; -----
REC_ORD  L      2
          =      PAR_NR_2      BM00.00 ; (PB (block trans) partner number)
          LD      161
          =D      OWN_BL_2      BM00.01 ; (PB (block trans) own block number)
          LD      41
          =D      PAR_BL_2      BM00.03 ; (PB (block trans) partner block no.)
          JPK     PB_REC , _____
                        PAR_NR_2 -|_____|- ORD_NR_2,
                        OWN_BL_2 -|_____|- PB_STA_2,
                        PAR_BL_2 -|_____|-
          L      PB_STA_2      BM00.06 ; (PB (block trans) job status)
          CMP     41                      ; job queue overflow?
          JP=     END                      ; yes -> repeat in next cycle
          JP      NXT_STEP                ; no -> status request
; -----
; Status request of receive job
; -----
REC_STA  JPK     PB_STAT , _____
                        ORD_NR_2 -|_____|- PB_STA_2
          L      PB_STA_2      BM00.06 ; (PB (block trans) job status)
          CMP     40                      ; receive job still being processed?
          JP=     END                      ; yes -> check again in next cycle
          CMP     14                      ; receive job completed?
          JP=     NXT_STEP                ; yes -> next job
          JP      S_ERROR                  ; else -> error analysis

```

The significance of communication errors depends on the application. In our example, we are counting the errors in a byte for later analysis.

```
; ***** Analysing Error Messages from the Status Request *****  
S_ERROR  NOP  
R_ERROR  INC      ER_CNT_2      BM00.15 ; (error counter (block trans) stn. 02)  
          JP      NXT_STEP      ; -> next job  
END      NOP
```

If you like, you can now transmit the program into the controller and test it using the various KUBES test functions.



You have learnt:

- how to define block transfer connections under VEBES
- how to program cyclic block transfer communication
- how KUBES modules PB_SEND, PB_REC and PB_STAT can be used for block transfer operations.

Exercise: "Block transfer on command" (single master system)

In this example we want to explain how block transfer tasks are handled on command. A master (stn. 1) communicates with a slave via block transfer operations. Sending and receiving are triggered on command.



The following example is principally structured like the previous one, but is furthered by the command for sending and receiving.

The complete example can be found on the enclosed diskette. The network is called "PB-BSP-4".

The KUAX 680S is equipped with 8 inputs and 8 outputs.



- The first step is to create a new network under VEBES.
- Take over the bus parameters of the FMS protocol..
- Define the following stations:

master 01	station address 1	device type KUAX 680I (or another master)
slave 02	station address 2	device type KUAX 680S
- Choose option "Connections - block transfer" in the "Network" menu. This opens the dialog box "Define block transfer connections".
- Click on the master in the "Source" field.
- Click on the corresponding "New" field. This opens the "Define block" dialog.



- Make the following entries:

OD index: 160

Operand: SBM00.00

Length of block (byte): 1



- Click on "OK". The dialog box disappears.
- Click on the slave on the "Destination" field.
- Click on "42" in the "OD index" box.
- Click on "Accept". The defined connection appears in the list "Existing connections" below.
- Use the following information to correspondingly define the connection from slave to master:
OD index slave: 41
OD index master: 161
Operand: SBM01.00
Length of block: 1
- Close the dialog box by clicking on "Exit".
- Change to KUBES "with project".
- Call up the Symbol Table Editor and enter the following:



Address	Symbol	Comment
I01.00	I_S_BLOC	"send block" command
I01.01	I_R_BLOC	"receive block" command
M00.00	S_BLOC	stored "send block" command
M00.01	R_BLOC	stored "receive block" command
BM00.00	PAR_NR_2	PB (block trans) partner number
BM00.01	OWN_BL_2	PB (block trans) own block number
BM00.03	PAR_BL_2	PB (block trans) partner block no.
BM00.05	ORD_NR_2	PB (block trans) job number
BM00.06	PB_STA_2	PB (block trans) job status
BM00.14	ST_CNT_2	step counter (block trans) stn. 02
BM00.15	ER_CNT_2	error counter (block trans) stn. 02
SBM00.00	S_BLC_02	block to be sent to station 02
SBM01.00	R_BLC_02	block to be received from stn. 02
PP00.01	PW00_00	edge analysis (pulse)
PP00.01	PW00_01	edge analysis (pulse)



- Quit the Symbol Table Editor.
- Open the Module Editor and load the organization module into the editor.

The main program controls the actual process and triggers block transfer communication.

- Write the following program or copy it from the preceding exercise.

Software

```
; ***** Main Program *****
MAIN    NOP
; processing block transfer
        JPP    BLCTRA02        5
; Writing 8 local inputs into 8 external outputs
        C1T8   I00.00
        =      S_BLC_02        SBM00.00 ; (block to be sent to station 02)
; Writing 8 external inputs into 8 local outputs
        L       R_BLC_02        SBM00.01 ; (block to be received from stn. 02)
        C8T1    O00.00
; *****
```



- Save the module and load program module "BLCTRA02" into the editor.

In our example we want to send a block to the 680S and also receive one block from the 680S.

Sending: master index 160 (SBM00.00) → 680S index 42 (8 outputs)

Receiving: master index 161 (SBM01.00) ← 680S index 41 (8 inputs)

Communication is not to take place cyclically, but upon command only:

Sending is triggered by input I01.00 and receiving by I01.01.



- Write the following program:



You can copy parts of the program from the previous exercise "cyclic block transfer".

Exercise: Block transfer on command

```
; ***** Block Transfer to/from Station 02 *****
; Commands for Sending/Receiving
; -----
; "Send block" command
; -----
      L      I_S_BLOC      I01.00 ; ("send block" command)
      =      PW00_00      PP00.00 ; (edge analysis (pulse))
      L      PW00_00      PP00.00 ; (edge analysis (pulse))
      S      S_BLOC      M00.00 ; (stored "send block" command)
; -----
; "Read block" command
; -----
      L      I_R_BLOC      I01.01 ; ("receive block" command)
      =      PW00_01      PP00.01 ; (edge analysis (pulse))
      L      PW00_01      PP00.01 ; (edge analysis (pulse))
      S      R_BLOC      M00.01 ; (stored "receive block" command)
; *****
```

The command is stored and reset after successful or unsuccessful execution.

Although these block transfer operations are controlled by commands, it still seems practical to structure it as a sequence chain. The reason for this is the fact that a slave can never process more than one communication task at a time. This situation is avoided with the sequence chain.

```
; ***** Step Control *****
; Communication problems with partner?
; -----
```

This request is only relevant for controllers with motor versions smaller 4.20.

As from version 4.20, the next three program lines must be removed.

```
; -----
      L      C_STAT02      PSa00.02 ; (communication status station 2)
      JP<>    CLR_STEP      ; error -> starting position
      NOP      ; no error -> continue
; -----
```

Software

```
; -----  
; Branch strip to the steps  
; -----  
STEPS    L        ST_CNT_2        BM00.14 ; (step counter (block trans) stn. 02)  
        CMP        0  
        JP=        NXT_STEP                ; -> next step  
        CMP        1  
        JP=        SND_ORD                ; -> process send job  
        CMP        2  
        JP=        SND_STA                ; -> status request send job  
        CMP        3  
        JP=        REC_ORD                ; -> process receive job  
        CMP        4  
        JP=        REC_STA                ; -> status request receive job  
        JP        CLR_STEP                ; -> reset step chain  
; -----  
; Set next step  
; -----  
NXT_STEP L        ST_CNT_2        BM00.14 ; (step counter (block trans) stn. 02)  
        CMP        4                ; arrived at last step?  
        JP>=        CLR_STEP                ; yes -> clear step counter  
        INC        ST_CNT_2        BM00.14 ; (step counter (block trans) stn. 02)  
        JP        END_STEP  
; -----  
; Reset step chain  
; -----  
CLR_STEP CLR        ST_CNT_2        BM00.14 ; (step counter (block trans) stn. 02)  
  
END_STEP JP        END
```

Exercise: Block transfer on command

```

; ***** Sending a Block *****
; -----
; Initiate send job
; -----

SND_ORD  L      S_BLOC          M00.00 ; (stored "send block" command)
        JPC     SEND            ; yes -> send
        L       3              ; no -> go to step 3 (receive job)
        =       ST_CNT_2        BM00.14 ; (step counter (block trans) stn. 02)
        JP      END

; Sending
SEND     L       2              ; partner number
        =       PAR_NR_2        BM00.00 ; (PB (block trans) partner number)
        LD      160
        =D      OWN_BL_2        BM00.01 ; (PB (block trans) own block number)
        LD      42
        =D      PAR_BL_2        BM00.03 ; (PB (block trans) partner block no.)
        JPK     PB_SEND , _____
                PAR_NR_2 -| _____| - ORD_NR_2,
                OWN_BL_2 -| _____| - PB_STA_2,
                PAR_BL_2 -| _____| -
        L       PB_STA_2        BM00.06 ; (PB (block trans) job status)
        CMP     41              ; job queue overflow?
        JP=     END            ; yes -> repeat in next cycle
        JP      NXT_STEP       ; no -> status request

; -----
; Status request of send job
; -----

SND_STA  JPK     PB_STAT , _____
                ORD_NR_2 -| _____| - PB_STA_2
        L       PB_STA_2        BM00.06 ; (PB (block trans) job status)
        CMP     40              ; send job still being processed?
        JP=     END            ; yes -> check again in next cycle
        =0      S_BLOC          M00.00 ; (stored "send block" command)
        CMP     7              ; send job completed?
        JP=     NXT_STEP       ; yes -> next job
        JP      S_ERROR        ; else -> error analysis

```

Software

```
; ***** Receiving a block *****
; -----
; Initiate receive job
; -----
REC_ORD  L      R_BLOC      M00.01 ; (stored "receive block" command)
          JPC      RECEIVE      ; yes -> receive
          L        0          ; no -> step chain into starting position
          =      ST_CNT_2      BM00.14 ; (step counter (block trans) stn. 02)
          JP      END
; Receiving
RECEIVE  L        2
          =      PAR_NR_2      BM00.00 ; (PB (block trans) partner number)
          LD      161
          =D     OWN_BL_2      BM00.01 ; (PB (block trans) own block number)
          LD      41
          =D     PAR_BL_2      BM00.03 ; (PB (block trans) partner block no.)
          JPK     PB_REC , _____
                   PAR_NR_2 -| _____| - ORD_NR_2,
                   OWN_BL_2 -| _____| - PB_STA_2,
                   PAR_BL_2 -| _____| -
          L      PB_STA_2      BM00.06 ; (PB (block trans) job status)
          CMP     41          ; job queue overflow?
          JP=     END          ; yes -> repeat in next cycle
          JP      NXT_STEP      ; no -> status request
; -----
; Status request of receive job
; -----
; Status request
REC_STA  JPK     PB_STAT , _____
                   ORD_NR_2 -| _____| - PB_STA_2
          L      PB_STA_2      BM00.06 ; (PB (block trans) job status)
          CMP     40          ; receive job still being processed?
          JP=     END          ; yes -> check again in next cycle
          =0     R_BLOC      M00.01 ; (stored "receive block" command)
          CMP     14          ; receive job completed?
          JP=     NXT_STEP      ; yes -> next job
          JP      R_ERROR      ; else -> error analysis
```

Exercise: Block transfer on command

The significance of communication errors depends on the application. In our example, we are counting the errors in a byte for analysis.

```
; ***** Analysing Error Messages from the Status Request *****  
S_ERROR  NOP  
R_ERROR  INC      ER_CNT_2      BM00.15 ; (error counter (block trans) stn. 02)  
          JP      NXT_STEP      ; -> next step  
END      NOP
```

If you like you can now transmit the program into the controller and test it using the various KUBES test functions.

4.3.4. Connecting the programming PC with the controller

There are three different possibilities of transferring the program into the controller. Which one is chosen depends on the interface via which the programming PC is connected to the controller to be programmed (see page 3-38). The "PLC" menu of the KUBES Main program menu contains the following menu items for establishing the connection between PC and controller:

- **Online V.24** for programming via the V.24 interface
- **Online PC** for programming via the PC-bus
(only for KUAX644)
- **Online PROFIBUS** for programming via the RS485 PROFIBUS interface (only works if the PC is equipped with a PROFIBUS PC card).

4.3.4.1. Online V.24

If the program is transferred via the V.24 interface, then the PC does not count as a PROFIBUS station. The PC is then only used for occasional communication with the controller to be programmed; it can take up no direct contact with the other PROFIBUS stations.

4.3.4.2. Online PC

This type of connection is the fastest possibility for programming. However, it only works between a PC and a KUAX 644. Again, the PC does not count as a PROFIBUS station in this type of programming. If you choose this menu item, the "Select online mode" dialog opens. Input the PROFIBUS station address for your controller. For the PC, use a station address which is not otherwise allocated to other network stations. Valid numbers are between 1 and 99. This station address is only assigned temporarily and is irrelevant for PROFIBUS network operation.

4.3.4.3. Online PROFIBUS

This connection is only possible if the programming PC has its own PROFIBUS interface. The PC then takes part in PROFIBUS communication as an individual station and it thus has its own station address. Choosing this option also calls up the "Select online mode" dialog. Enter the PROFIBUS station addresses for the PC and the controller as they have been defined by VEBES and set on the device.

You can enter the station addresses of PC and controller also through option "PROFIBUS addresses". The controller to be programmed will go online after you have input the station addresses. You can now transfer the program.



The entered station addresses must correspond to those defined under VEBES and to the DIP switch settings on the actual device.

4.3.5. Using KUBES to monitor the PROFIBUS network

KUBES provides you with a large number of test functions which you can use to monitor the program run:

- Dynamic display in the Module Editor
- "Test" menu of the Module Editor
- Display address range
- Display single address
- Analog diagram
- Logic diagram

The display functions for address ranges and single addresses are particularly well-suited for monitoring the PROFIBUS network. The Logic diagram and the Dynamic display of the module editor may be the function of your choice from time to time.

4.3.5.1. Display address range

If you are working with a network project, KUBES has extended the range selection dialog of the "Display address range" option by the entries "PROFIBUS messages" and "PROFIBUS". This is identical to the changes made to the symbol table in this case. The PROFIBUS messages will be explained in another chapter (page 4-62). They can be displayed in the corresponding operand ranges shown in the address range display.

If you select the "PROFIBUS" range, the program will display an overview of all stations networking with the controller via the PROFIBUS and maintaining a process chart connection to the controller. Selecting one of these stations displays its operands (which are external operands from the controller's point of view). You can now use the address range display to view the state of these operands.

4.3.5.2. Display single address

Via the "Display single address" dialog, you have the possibility to monitor up to 16 operands simultaneously. External operands can be monitored in exactly the same way as local ones. To do so, simply enter the descriptive operand names into the various "Address" fields. After confirming the entry, the current value of the operand is read in from the controller. The display can also be dynamic. Apart from displaying local and external operands, you can of course use the "Display single address" to also read the PROFIBUS messages (see page 4-62).

4.3.5.3. Dynamic display in the Module Editor

The "Dynamic display" is one of the KUBES test functions. It allows you to monitor the signal states of all operands visible within the editor window. It is particularly suitable for testing individual parts of the program.

4.3.5.4. Logic diagram

Use the logic diagram to record the course in time of bit variables. For byte variables, the program analyses bit 7. Up to 8 optional variables can be chosen. These may include internal or external operands. Enter the addresses of the desired operands into the corresponding text fields. You can start (or stop) the recording either with a trigger or through menu option "Start - Immediately" or "Stop - Immediately". There are several recording speeds to choose from.

4.3.6. PROFIBUS messages

PROFIBUS messages are shown in the following operand ranges:

- PFaxx.yy PROFIBUS error messages
- PSaxx.yy PROFIBUS status messages
- PEaxx.yy PROFIBUS event notifications

The PROFIBUS messages are valid for both the FMS and the DP protocols.

Operand range PFaxx.yy is reserved for messages concerning bus failures or any other faults occurring.

Status messages are all messages concerned with the status of the communication partners, for which a connection has been defined under VEBES. An example of a status message is "Fatal connection failure". Status messages are read-only notifications.

Event notifications are transmitted by the individual communication partners. They concern current events such as "short circuit at output". They are transmitted with high priority to each controller which has a defined connection (under VEBES) to the station in question. Event notifications are only sent out once at the first occurrence of the event.

The messages are represented by code numbers, whose significance is explained in the following tables.

4.3.6.1. PROFIBUS error messages



Messages containing information about the **ALI** operating state (application layer interface), certain objects and errors are shown in operand range **PFaxx.yy**. The ALI operating state is reported in operand **PFa00.00**. If necessary, an error number dependent on the ALI operating state is shown in operand **PFa00.01**. The status of the objects is indicated in operand **PFa00.07**.

PROFIBUS error messages

Operand		Description
PFa00.00		ALI operating state (see table on page 4-63)
PFa00.01		current error number, depending on ALI operating state: (see table on page 4-64)
PFa00.04		type of object: 2 = process map object 3 = block transfer object
PFa00.05	Low Byte	index of the object whose status is shown in PF00.07
PFa00.06	High Byte	
PFa00.07		object status (see table on page 4-65)
PFa00.08 to PFa00.12		no function at present
PFa00.13		255 = connection error
PFa00.14		255 = event notification received
PFa00.15		255 = process map error

ALI operating state (application layer interface)

Operand PFA00.00

Value	Description
255	undefined status, ALI has not been activated
0	ALI has been activated and there has been no reset
1	ALI RESET condition
2	successful PCB reset but no parameterization yet
3	error, unsuccessful reset (see table on page 4-64)
4	transfer of parameters to ALI o.k.
5	initialisation of the application layer o.k.
6	error in parameterization of process map or domain (s. table on page 4-64)
7	error in parameterization of CRL (see table on page 4-64)
8	ALI operating. It has initiated the connections and is starting to process the process map and is ready to start block transfer operations.
9	serious internal error

Error codes depending on the ALI operating state (errors 3, 6 and 7) in Operand PFa00.01

Value	Description
Error messages with ALI status = 3:	
0	OK, successful reset
1	timeout for mailboxes to the application layer
2	reset not processed by FMS
3	reset not acknowledged by FMS
4	address used twice on the bus
5	FMA_SetValue timeout. Error of the basic parameters
6	no FMS acknowledge after reading back SW release
7	hardware failure in the transceiver
8	internal error; no free processing block
9	no FMS answer-back message after setting the basic parameters
10	wrong FMS answer-back message after setting the basic parameters
11	wrong Poll_SAP entry into CRL
Error messages with ALI status = 6:	
0	OK
1	object number greater than number of declared objects
2	object number smaller 1
3	object description exists
4	object index used elsewhere
5	update-time outside the permissible range
6	block transfer object longer than 64 KByte
7	no index smaller 17 allowed
8	index for process map object exceeding permissible range
9	impermissible object type
10	partner address identical to own station address
11	index for block transfer object exceeding permissible range
12	CR outside the permissible range
13	LSAP parameterized already
14	CR allocated already
15	no more than 8 parallel services allowed
16	number of extended CRL entries too large
17	numbers indicated in global configuration parameters too large
18	impermissible CRL type
Error messages with ALI status = 7:	
[CR]	wrong CR number (see VEBES network listing)

Status of an object (operand PFa00.07)

Value	Description
0	normal status (OK); no indication with special order and event objects
1	indication
2	call-up in combination with special orders and after ALI startup
3	no partner reaction (timeout)
4	object does not exist in the partner
5	object access impossible (input/output reversed)
6	inconsistent data length
7	no CR available
8	connection breakdown
9	reject message received
10	unexpected negative response received
11	object configuration error (wrong initialisation parameters)
12	no description
13	send data requested (downloading)
14	
15	send-wait response (downloading)
16	
17	receive-wait response (uploading)
18	receive data requested
19	receive data transferred
20	wait download con
21	wait download seq ind
22	download wait data
23	wait term download ind
24	wait download seq con
25	wait term download con
26	
27	wait ini upload con
28	wait upload seq con
29	wait term upload con
30	wait upload seq ind
31	upload wait data
32	wait term upload ind

4.3.6.2. PROFIBUS status messages

In operand range PSa00.00....07.15, there is one byte available for each potential PROFIBUS station according to the table below. This byte (per station) is reserved for indicating the status of the corresponding station by a number whose significance is explained in the second table.

Allocation of operand range PSa00.00-PSa07.15 to the PROFIBUS stations

Operand	Station address
PSa 00. 00	0
01	1
to	
15	15
01. 00	16
01	17
to	
15	31
to	
07. 00	112
01	113
to	
14	126

Significance of the partner status codes

Value	Description
0	normal communication status, connection OK
1	unsuccessful initialisation of connection
2	connection breakdown
3	-
4	no connection defined

4.3.6.3. PROFIBUS event notifications

In operand range PEa00.00....PEa07.15, there is one byte available for each PROFIBUS station according to the table below. This byte (per station) is reserved for indicating alarm messages of the corresponding station by a number whose significance is explained in the second table.



The displayed messages are only valid if the partner status message of that station is "0".

Event notifications are individually sent to the connected stations. Because of this, it may take a certain amount of time until the message has reached every network partner. If a new alarm occurs in the meantime, an internal priority control decides if this one should precede the one before or not (small number = high priority).



A controller in RESET mode retains its connections. It can thus continue to receive alarm messages.

Allocation of operand range PEa00.00-PEa07.15 to the PROFIBUS stations

Operand	Station address
PEa 00. 00	0
01	1
to	
15	15
01. 00	16
01	17
to	
15	31
to	
07. 00	112
01	113
to	
14	126

List of the event notifications put out by Kuhnke devices

Value	Priority	Description
0	-	no event message (not transmitted)
1	3	short circuit at output
2	5	undervoltage or overvoltage
3	2	watchdog triggered and cause removed (KUAX 680S only)
4	6	timeout - connection aborted
5		error message not transferrable via PROFIBUS
6		
7	7	wrong modules plugged in (KUAX 680S only)
8	1	checksum error in user program (controller only)
9	4	hierarchy error
11		unused
10	6	no memory module plugged in (not in KUAX 644)
12	3	short circuit (error #1) removed
13	5	undervoltage (error #2) removed
.		
.		
.		
124	8	controller will be started by KUBES (RUN)
125	8	controller will be stopped by KUBES (STOP)
126	8	controller will be reset by KUBES or by command (RESET)
127	8	controller will be put into test operation by KUBES
128 bis 255	9	event notification defined and created by the user; the user program writes the event notification into the operand (PEaxx.yy) that corresponds to the station's own station number.



Event notifications by devices sold by other manufacturers are not listed in this table as they are not known. Please refer to the relevant manufacturers' data for the necessary information.

4.3.7. Monitoring the bus operation via the user program

It is necessary in practice to monitor the bus operation through the user program so that suitable measures are automatically taken in the case of an error.

Security is very important in this context depending on the type of application. In some circumstances, individual machines or sub-systems must be turned off or reset to avoid danger for man and machine.

Bus monitoring by the program can be divided into 2 sections:

- bus control at startup
- bus control during operation



Bus control at startup first reads the **ALI** operating state and the PROFIBUS status of the individual communication partners. Only then is communication possible. In single-master networks with one or several slaves it may be practical to skip the main program until this startup routine has been completed.

For bus control during operation, the ALI operating state is analysed continuously. Furthermore, communication with the other stations is monitored and possible event notifications from the individual stations are read. Operational bus control is also responsible for initiating application-dependent reactions to errors.

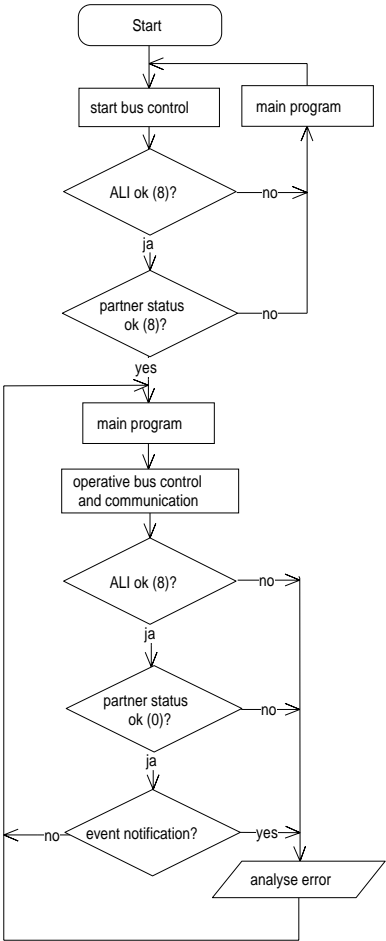
Suggestion #1:

The main program is only started after the bus has been established completely. This structure is suitable in cases where a start without a connection to the partners is impractical (e.g. in single master systems). Refer to example "PB_BSP_2" for a realization of this kind of setup (see page 71 and following).

Suggestion #2:

In this case, the main program will always run, even when the bus has not yet been fully established. Only the communication tasks (e.g. block transfers) will be held back during that time (see example program "PB_BSP_3" or "PB_BSP_4" on the diskette). This structure is practical for all cases where the main program should also run without any communication processes (e.g. in multi master systems, where only information but no control signals are transferred).

This type of setup is represented in the following diagram:

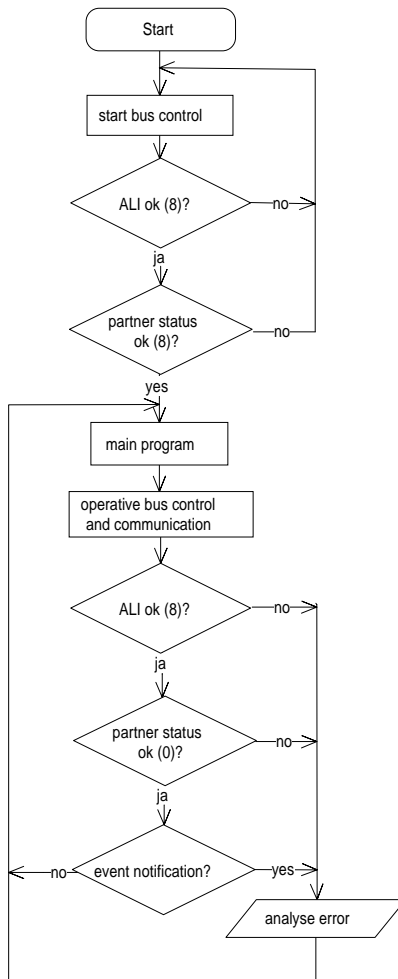


Exercise: Bus control in the master

With this exercise we want to illustrate the programming of the bus monitoring operations. The flow chart below is an overview of the program structure according to suggestion #1 (see page 4-69).



The whole example is on the enclosed example diskette. The network is called "PB_BSP_2".



The KUAX 680S is equipped with 8 inputs and 8 outputs.



- The first step is to create a new network.
- Take over the bus parameters of the FMS protocol.
- Define the following stations:
master01 stn. address 1 type of device: KUAX 680I (or another master)
slave02 stn. address 2 type of device: KUAX 680S
- Define the process chart connections between master and slave (and vice versa).
- Change to KUBES "with project".



- Make the following entries into the symbol table:

Address	Symbol	Comment
LM00.00	BUSSTART	bus has been started
LM00.02	C_FAIL02	serious communic. error stn. 2
LBM00.02	WD_002	watchdog in station 2
PEa00.02	EVENT_02	event notification station 2
PSa00.02	C_STAT02	communication status station 2
PFa00.00	ALI_STAT	ALI operating state

Reminder: you get to operand ranges PSa, PFa and PEa via option "PROFIBUS messages" of menu item "Select" of the "Selection" menu of the Symbol Table Editor.

- Load the organization module into the Module Editor.

After the start, the main program will be skipped until the bus has been set up completely.



- Write the following program:

```
; ***** Bus Control *****
BUS_CTRL JPP      BUSCONTR          1
          L        BUSSTART          LM00.00 ; (bus has been started)
          JPCN     END                ; no -> skip main program
; *****
```

This is the start of the main program that is used for controlling the actual process. Into our example, we have only included a few commands for reading from and writing into external inputs and outputs.

```
; ***** Main Program *****
MAIN      NOP
; Read local input and write into external output
          L        I00.00
          AN        C_FAIL02          LM00.02 ; (serious communic. error station 02)
          =        O02a00.0
; Read external input and write into local output
          L        I02a00.0
          AN        C_FAIL02          LM00.02 ; (serious communic. error station 02)
          =        O00.00

; End of main program
END      NOP
; *****
```



- Save the module and load program module „BUSCONTR“ into the Module Editor.
- Write the following program:

The bus control at startup is initiated after starting the master. The main program must not start until the bus is ok (see ORG module).

```

; ***** Start Bus Control *****
STRCTRL L      BUSSTART      LM00.00 ; (bus has been started)
          JPC      CONTCTRL          ; yes -> operational bus control

; -----
; ALI status ok?
; -----
          L      ALI_STAT      PfA00.00 ; (ALI operating state)
          CMP     8              ; ok?
          JP<>    NO_START      ; no -> do not allow start

; -----
; PROFIBUS status of the communication partners ok?
; -----
          L      C_STAT02      PSa00.02 ; (communication status station 2)
;          0      <more communication partners?>
          CMP     0              ; ok?
          JP<>    NO_START      ; no -> do not allow start

; -----
; Conditions for start fulfilled => enable main program
; -----
OK_START =1      BUSSTART      LM00.00 ; (bus has been started)
          JP      CONTCTRL

; -----
; Conditions for start not fulfilled => do not enable main program
; -----
NO_START =0      BUSSTART      LM00.00 ; (bus has been started)
          JP      END

```

Operational bus control is a permanent bus control during operation. This part of the program will be permanently run through after the bus has been started and the main program enabled (see above). Here the reactions to bus failures differ

from those at the start, as a PLC program must not be skipped just kike that once it has been started. Safety plays an important role, depending on the type of application. There is a whole range of failure treatments from simply ignoring it or indicating/displaying the error or switching off individual parts of a machine/plant or even to resetting the controller (RESET command). In our example we are setting a failure marker for every partner and connect this marker in the main program to the external I/Os ("AN").

```
; ***** Operational Bus Control *****
; Checking the ALI status
; -----
CONCTRL L      ALI_STAT      PFa00.00 ; (ALI operating state)
          CMP      8          ; ALI successfully initialised?
          JP<>    BUS_FAIL    ; yes -> failure treatment bus failure

; -----
; Checking station 02 for proper communication and possible events
; -----
          JPP      COMCTR02    2
          As long as a station is communicating with further stations, the
          communication and event examination of these stations can be
          called up from here (see example on the next line).

; !!!      JPP      COMCTR__
          JP      END

; -----
; Treatment of general types of bus failures
; -----
BUS_FAIL =1    C_FAIL02      LM00.02 ; (serious communic. error station 02)
; !!!      =1    C_FAL__      <other stations, if exist>
          JP      END

; -----
; End of bus control
; -----
END      NOP
; *****
```



The error markers will only be reset when no failures have been reported in module COMCTRxx (see there).



- Load program module COMCTR02 into the Module Editor.
If DIP switch 9 of the 680S is "OFF" the controller will switch off all outputs if a bus error occurs.

- Write the following program:

```
; ***** Connection Monitoring Station 02 *****
; -----
; Communication error? (partner status)
; Note: If Dip switch 9 of the 680S is "OFF" the controller will switch all
;       outputs off when a bus error occurs.
; -----
PS_02  L      C_STAT02      PSa00.02 ; (communication status station 2)
        CMP    0              ; connection ok?
        JP=    EVCTR_02       ; yes -> event control
        CMP    1              ; connection unsuccessfully initiated?
        JP=    PS_FAIL        ; yes -> failure treatment bus failure
        CMP    2              ; connection collapsed?
        JP=    PS_FAIL        ; yes -> failure treatment bus failure
        CMP    4              ; connection defined under VEBES?
        JP=    END            ; no -> end of connection monitoring
        JP     PS_FAIL        ; else -> failure treatment bus failure
```

The following program sections analyse the event notifications of the communication partner. In our example, we are using the messages of the KUAX 680I. Please take into consideration that other devices may dispatch different messages.

Exercise: Bus control in the master

```

; ***** Error Messages of Station 02 *****
; -----
; Error message? (Event)
; -----
EVCTR_02 L      EVENT_02      PEa00.02 ; (event notification station 2)
        CMP      0              ; Event?
        JP=      NO_FAIL        ; no -> go to end

; -----
; Short circuit (message 1)
; The 680S switches all outputs off if a short circuit occurs at one output.
; -----
EVENT1 L      EVENT_02      PEa00.02 ; (event notification station 2)
        CMP=      1
        JPCN      EVENT2
        JP      EVN_FAIL        ; -> set error marker

; -----
; Undervoltage (message 2)
; If Dip switch 10 of the 680S is "OFF", the controller will deactivate all out-
; puts when undervoltage occurs. Inputs will still be read and sent correctly.
; -----
EVENT2 L      EVENT_02      PEa00.02 ; (event notification station 2)
        CMP=      2
        JPCN      EVENT13
        JP      EVN_FAIL        ; -> set error marker

; -----
; Voltage back to normal (message 13)
; -----
EVENT13 L      EVENT_02      PEa00.02 ; (event notification station 2)
        CMP=      13
        JPCN      EVENT3
        CLR      EVENT_02      PEa00.02 ; (event notification station 2)
        JP      NO_FAIL        ; -> go to end

```

Software

```
; -----
; WATCHDOG (message 3)
; The 680S only puts out this message if a watchdog has been removed again by
; internal reset. The message can thus only be analysed as information in the
; master (e.g. increment byte).
; If the watchdog cannot be removed, the master assumes the occurrence of a
; communication error (PSaxx.yy = 2) after a while and reacts accordingly.
; -----
EVENT3  L      EVENT_02      PEa00.02 ; (event notification station 2)
        CMP=    3
        JPCN    EVENT5
        INC     WD_002      LBM00.02 ; (watchdog in staton 2)
        CLR     EVENT_02    PEa00.02 ; (event notification station 2)
        JP      END

; -----
; Bus failure (message 5)
; This is only listed here for reasons of completeness. Communication errors are
; otherwise recognized as partner status errors (PSa00.02 = 2) already.
; -----
EVENT5  L      EVENT_02      PEa00.02 ; (event notification station 2)
        JP      EVENT7

; -----
; Wrong module configuration (message 7)
; This message can have 2 different causes:
; 1. A module is in the 680S that is not designed for this controller.
; 2. There are more or less modules in the 680S than were defined under VEBES.
; In neither of the two cases can reading from or writing into the inputs and
; outputs be guaranteed to fully coincide with the user program instructions.
; -----
EVENT7  L      EVENT_02      PEa00.02 ; (event notification station 2)
        CMP=    7
        JPCN    END
        JP      EVN_FAIL
```

PROFIBUS status messages have priority over event notifications. Event notifications thus become invalid and must be cleared.

```
; ***** Error Analysis *****
; Error analysis PROFIBUS status
; -----
PS_FAIL CLR      EVENT_02      PEa00.02 ; (event notification station 2)
        =1      C_FAIL02      LM00.02 ; (serious communic. error station 02)
        JP      END

; -----
; Error analysis Event
; -----
EVN_FAIL =1      C_FAIL02      LM00.02 ; (serious communic. error station 02)
        JP      END

; -----
; No error
; -----
NO_FAIL  =0      C_FAIL02      LM00.02 ; (serious communic. error station 02)
        JP      END

END      NOP
```

If you like you can now transfer the program into the controller and use the KUBES test functions to test its integrity.

A. Glossary

Application Layer Interface (ALI)

ALI interfaces between the application layer (layer #7) of the PROFIBUS-FMS and the real application process. In Kuhnke controllers it is a part of the monitor program. ALI allows access to the services of the PROFIBUS application layer and provides the communication functions required by the application process. It also maps the Virtual Field Device (q.v.) onto the real field device. It includes service management and object administration.

Application process

A production process which is controlled by automation devices. Under the aspect of communication, application processes also comprise all other programs and tasks, which cannot be allocated to a communication layer such as operating systems or communication drivers.

Bit times

A bit time is the time needed to transmit a bit. This time directly depends on the set baudrate.

Block transfer

Under the management of VEBES, PROFIBUS communication also allows for the transmission of data blocks which are not to be sent via the process chart. This type of communication is called block transfer. It mainly serves transferring large amounts of data, which are not to be transmitted or received cyclically but on command. Block transfer consists of transmitting the data of one block (own block number) into the block of the communication partner (partner block number) Both data blocks must have the same length.

Under VEBES you can define up to 10 data blocks per station: blocks no. 160..169 (corresponding to OV indices 160..169). These are operand ranges whose starting address and length are user-defined.

Appendix

Communication via block transfer is only necessary in connection with certain devices. The actual block transfer jobs are handled by KUBES modules in the user program of the relevant controllers.

Broadcast message

A bus station transmits messages not only to one other communication partner but to all other stations. This type of message is called a broadcast message.

Bus

In transmission technology, a bus is defined as a shared transmission medium for the messages of all bus stations, e.g. the cable. Field buses and process buses are implemented to transmit the data needed for automatic control of production processes.

Bus access protection

PROFIBUS provides access protection mechanisms for objects and communication endpoints to ensure the security of the equipment. Objects are protected by entering them into the object dictionary. For certain objects, only certain services (e.g. read only) may be allowed.

Communication endpoints are protected via entries in the communication relationship list which only give certain communication partners access to certain endpoints.

With KUBES you also have the possibility to protect network projects with a password (Project menu, option Password).

Bus access right

The right of the masters to access the PROFIBUS is determined by a “token passing” procedure. A token is a certain bit sequence, which is passed on in a logical ring between the active stations of the network. Only the station with the token has access to the bus. The token must be passed on to the next station within a set time interval. Slaves never receive the token.

Bus parameters

There are two groups of PROFIBUS operating parameters: the parameters of one group are the same for all stations while the parameters of the other group are only valid for one specific station. The parameters of the latter group depend on the PROFIBUS network stations where they are also set.

The following bus parameters are defined under VEBES:

- Target Rotation Time (time in which all stations with access to the bus should have possessed the token once)
- GAP Update factor (time interval during which the network is checked for any new bus stations)
- transmission rate (in kbaud)
- Slot Time (time between transmission of a request and arrival of the corresponding response or acknowledgement)
- min. Station Delay Responder (min. response time of the network stations)
- max. Station Delay Responder (max. response time of the network stations)

CI Cyclic control interval

The cyclic control interval is a bus monitor function. It is a cyclic interval to check the bus for existing connections at times when no user data is being transferred via the bus.

Communication endpoint

The communication endpoints define the logical connection of the communication relationship between two application processes by establishing both ends of the relationship. There can be more than one communication relationships between two application processes. These can be clearly identified and differentiated by the defined endpoints. In the communication relationship list, the communication endpoints are defined by the corresponding communication reference.

Communication reference (CR)

The communication reference is marked by the addresses of the communication endpoints, with which application processes are integrated in the PROFIBUS communication system. Communication references are device-specific. They are defined by the network users and stored in the communication relationship list.

VEBES automatically assigns communication references. They can be viewed on the printout of the communication relationship list.

Communication relationship

A communication relationship is the logical communication channel between two communication partners. PROFIBUS communication can be either connectionless or connection-orientated communication. Connection-orientated communication relationships have transmission channels in both directions of the communication partners so that a message can be confirmed by a response transmitted via a return channel. The connections of such communication relationships must be initialized in a connection startup phase and must be released again after data transmission. While only requiring one logical channel to each communication partner, connectionless communication relationships do not allow confirmation responses to a message. They are therefore used for (unconfirmed) multicast or broadcast services.

The communication relationships are defined during the project planning phase and stored in the communication relationship list.

Communication Relationship List (CRL)

The communication relationship list contains information about the connections planned for the network project. This information is stored independent of its later application and includes items such as communication reference, source SAP, connection type, connection attribute, remote address.

Every communication partner has such a list for his communication relationships. Every communication partner can have up to 63 communication relationships. These are identified by the communication reference. A separate communication relationship has to be reserved for multicast and broadcast messages.

Connection

Connection-orientated communication relationships fall into the two categories of 'open' and 'defined' connections. The difference between the two categories is the point in time at which the data link layer (layer 2) addresses for the connection are determined. Defined connections are set during the project planning and startup phase so that access protection exists during the initialization phase already. Open connections are only entered into the CRL when the connection is being established. After that they behave like defined connections, however.

The connection type is entered into the communication relationship list with the connection attribute.

Connection attribute

In the communication relationship list, the connection attribute characterises the type of connection. VEBES differentiates between open connections of the responder (O), open connections of the initiator (I), and defined connections (D).

Open connections of the responder (O):

The connection attribute is allocated to a specific communication reference (CR). Via an "O"-classified communication reference, a station can be addressed at any time by any station. Thus every connection establishment requested by another station via this CR will be accepted as long as the corresponding communication channel has not already been occupied by another station.

Open connections of the initiator (I):

Like defined connections, I-connections are always one-to-one connections between two partners. It is possible, however, to use the same SAP (service access point, q.v.) for handling another I-connection at another time.

I-connections can only be maintained between masters.

Defined connections (D):

A defined connection is a set permanent connection between two stations. It is set up during bus configuration. The corresponding SAPs of the two communication partners are only used for this CR.

Connection types

The connection type contains a definition of the communication partners and the form of communication.

VEBES manages the following connection types:

- MMAC master-master connection with acyclic data transmission
- MSAC master-slave connection with acyclic data transmission
- MSAC_SI master-slave connection with acyclic data transmission and slave initiative

Context management services

Context management services allow initiating, releasing and aborting connections. They also allow rejecting illegal services.

Context management services include:

- initiate (initiates the connection between two communication partners)
- abort (releases an existing connection between two communication partners)
- reject (rejects illegal services).

Data transfer services (data transmission)

PROFIBUS supports the two basic data transfer services of cyclic and acyclic data transmission. Cyclic data transfer signifies fast, recurring operations in which data is requested from the slaves by a master to give the master an up-to-date process chart of the current technical process. Acyclic data transfer signifies data requests only when they are needed. Kuhnke PROFIBUS networks only support acyclic data transfer services.

Data types

PROFIBUS provides the following standard data types to define the structure of simple variable objects:

- boolean
- integer (8, 16, 32 bits)
- unsigned (8, 16, 32 bits)
- floating point (IEEE std. 754 - short real number with 32 bits)
- octet string (binary encoding)
- visible string (ISO 646)
- date (calendar date and time)
- TimeOfDay (milliseconds elapsed since midnight)
- TimeDifference (current time in milliseconds)
- bit string (as octet units)

VEBES allows defining data types integer (8, 16, 32 bits), unsigned (8, 16, 32 bits) and octet string.

Decentralized periphery

see 'Field device' and 'PROFIBUS protocols'

DP protocol

see 'PROFIBUS protocols'

Field device

Field devices are all devices which are connected directly to the physical process (process peripheral devices), e.g. sensors, actuators, bar-code readers, measuring systems.

FMS (Fieldbus Message Specification)

As a sub-layer of layer 7 (application layer) of the PROFIBUS architecture (see ISO/OSI communication model), the purpose of the FMS is to make sure that all communication partners observe the rules for exchanging protocol data units. The FMS also generates data packages of the information to be transmitted via a requested service, encodes these packages and decodes incoming messages.

FMS protocol

see 'PROFIBUS protocols'

GAP Update factor

see 'Bus parameters'

Initialization / startup

Initialization is the startup phase of a system. During initialization, hardware and software take on their respective operating conditions.

ISO/OSI communication model

This communication model is a reference model (OSI = Open System Interchange) of the International Standardization Organization for communication systems. Communication processes are divided into 7 layers. Each layer makes certain services available at the interface to the next communication level. The layers follow each other in a hierarchical order. Each layer can only communicate with the neighbouring layer.

Messages run along logical pathways through the individual layers so that they are transferred without any loss or change of information.

The model is made up of the following layers:

1. physical layer
2. data link layer
3. network layer
4. transport layer
5. session layer
6. presentation layer
7. application layer

The PROFIBUS FMS architecture is based upon this model and is sub-divided into the three following layers: layer 1 = bit transfer layer (PHY), layer 2 = fieldbus data link layer (FDL), layer 7 = application layer (as sublayers FMS and LLI with the application layer interface and the lower layer interface).

KUBES

KUBES is a programming and user program for Kuhnke PLCs. It provides all functions for writing, commissioning, optimizing and documenting control programs.

It allows programming and monitoring controllers networking with PROFIBUS.

KUBES Modules

KUBES modules are special-solution modules, e.g. for block transfer communication operations under VEBES. They are written by Kuhnke, either in a high-level programming language or Assembler, and delivered in one or more module libraries on diskettes. By setting various input and output parameters, it is possible to carry out their function with different variables. The modules can be implemented several times into the same program using various parameters.

Appendix

There are three KUBES modules available for block transfer:

- PB_SEND is used for transmitting a data block to the communication partner
- PB_REC is used for receiving a data block from the communication partner
- PB_STAT is used for requesting the status of a job.

These KUBES modules contain several or all of the following parameters:

- Partner station address. Address of the communication partner to whom the data is transmitted or from whom it is requested.
- Own block number. Block number of the data block defined in one's own station.
- Partner block number. Block number of the data block defined in the communication partner's station.

Job number. The KUBES module (PB_SEND or PB_REC) writes an identification number into "job number" when placing the job. The ID no. is necessary for further processing of the job.

- Status. The KUBES module writes an identification number into "status" as information about the status of the specified job.

Master

In communication relationships, masters are active stations, i.e. they are clients or service requesters. In PROFIBUS networks, for example, a master can be a programmable logic controller (PLC). Masters actively access the bus and transmit messages. In PROFIBUS networks, the connected masters use a token to define which one of them currently has right of access to the bus. Service providers are passive stations. They are called slaves or servers.

Modu Control KUAX 657P

The KUAX 657P is a modularly constructed programmable logic controller made by Kuhnke. The rack is available with 4, 8 or 19 free slots. The KUAX 657P can be equipped with various function modules such as fast counters, stepping motor

control, axle positioning, temperature regulator, and a PROFIBUS module. A KUAX 657 P that equipped with a PROFIBUS module can be integrated into PROFIBUS networks where it works as a master.

Multicast message

A multicast message is a message which is transmitted by one network station to a group of several communication partners. Although this group consists of more than one station, their number and addresses are defined. A message which is transmitted to all other bus stations, on the other hand, is called a broadcast message.

Network project

VEBES is the network configurator software to create network projects for controllers networking via PROFIBUS. Controllers KUAX 644, KUAX 657P and KUAX 680I are networkable PLCs of the Kuhnke product range. Controllers KUAX 657 and KUAX 667 have no PROFIBUS interfaces. They are thus not suitable for implementation in PROFIBUS networks.

Object

All data of the communication partners (measuring values, programs, events...) which is part of an application process and is to be exchanged between the stations is defined as objects. Objects have a certain structure and are equipped with attributes (data type, access right). Objects can consist of data fields (inputs, outputs, markers), text messages, simple variables or combinations of these. The objects of a communication process are defined and stored in the object dictionary.

Object code

The object code is used by VEBES to identify the object type as simple variable, array or event.

Object dictionary (OD)

A dictionary or directory of all objects taking part in a communication process. All data concerning one object are combined in one line which is preceded by an index. The object description is defined by the network station where the real object exists (Source OD). The other network stations keep a partial or complete copy of the object descriptions of their communication partners (Remote OD).

Object dictionary index (OD index)

The OD index is the logic addressing of an object. It can be transmitted via the bus with a message only a few octets long and is defined in the object dictionary of the corresponding device.

Object type

VEBES supports the following types of communication objects:

- Simple variables are non-divisible objects (measuring values, times, device status). Their structure is defined by the data type.
- Array is a series of similar elements of the type 'simple variable'. The data type depends on the contained elements.
- Event (alarm) contains an important message of a defined data type. The message is transmitted to the communication partners with high priority.

PROFIBUS also employs the following object types:

- Record is a series of elements of the type 'simple variable' which are not similar to each other.
- Domain is a logically connected memory range of defined length. It can contain data or programs. Its data type is always octet string.

OD management services

OD management includes services for reading and writing the object dictionaries of the virtual field devices (VFD) of the communication partners. The following services are included:

- Get OD (reading the object descriptions)
- Initiate Put OD (initiating the put OD service)
- Put OD (writing of object descriptions into object dictionaries)
- Terminate Put OD (terminating the put OD service)

PC Control KUAX 644

The KUAX 644 is a powerful programmable logic controller made by Kuhnke. It can be plugged into an 8-bit slot on any IBM-compatible PC. It is equipped with a PROFIBUS interface and can thus be used as a master in network systems. Although the KUAX 644 has no inputs and outputs of its own it can read and process a large number of signals by cooperating with decentralized input/output devices such as the KUAX 680S. Its most prominent feature is the very fast data transfer between PC and PLC via the parallel PC bus.

Process chart

The process chart is a memory area where the status of all external operands (i.e. the inputs and outputs to or from the connected communication partners) is stored as it is at a defined point in time, for example at the end of a program cycle. Changes in the signal status cannot be taken into account while the next program cycle is being processed. The user program treats external operands like inputs and outputs.

PROFIBUS

The word PROFIBUS is an acronym of PRocess Field BUS. It signifies an open bus system for network systems at the field level. Open means that devices of different manufacturers can communicate with one another in accordance with the OSI model (Open System Interchange) of the ISO (International Standardization Organization).

PROFIBUS FMS

see 'PROFIBUS protocols'

PROFIBUS protocols

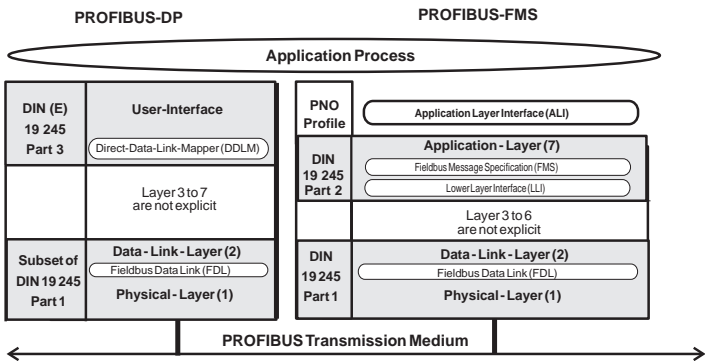
Kuhnke PROFIBUS networks can be operated with two different PROFIBUS protocols: PROFIBUS FMS and SINEC L2-DP. The PROFIBUS FMS protocol corresponds to the standard as described in DIN 19245, Parts 1 and 2. The FMS services open a wide field of possible applications. They are very flexible and capable of managing extensive communication tasks.

The SINEC L2-DP protocol is a newer PROFIBUS variant based on Part 1 and the German draft standard DIN 19245, Part 3. This PROFIBUS variant was especially designed for fast data exchange between PLCs and decentralized peripheral devices (DP stands for Decentralized Periphery). As opposed to systems running the FMS protocol, the SINEC L2 -DP protocol only provides services for mono-master systems with up to 30 slaves. One positive aspect of this reduced network size is the reaction times of this bus which are drastically shorter than in FMS-operated networks. Kuhnke devices which can be integrated into DP buses as slaves carry a 'DP' after their name (e.g. 680S DP).

There are differences in the architecture of the two protocols:

In both of them layers 1 and 2 of the ISO/OSI reference model have been realised, i.e. both protocols use the same bus access protocol and the same transmission technique. While the FMS protocol also provides layer 7 services, this layer does not figure in the DP protocol. There is a user interface instead, which has access to layer 2 through its integrated Direct Data Link Mapper (DDLM). The user interface provides all user application functions and definitions of system and device behaviour of the DP device types. The DP protocol has no ALI (Application Layer Interface) which, in the FMS protocol, forms the interface between the application process and layer 7 (application layer) of the OSI reference model.

For an overview of the architecture of the two protocols see the illustration on the next page.



Architecture of the PROFIBUS FMS and PROFIBUS DP protocols

Profi Control KUAX 680I

The KUAX 680I is a powerful mini-controller with a modular design. Kuhnke deliver this PLC equipped with 4 or 8 module slots and one or two PROFIBUS interfaces. There are connectors for three-wire connectors, so that proximity switches and similar extension can be supplied via the same cable as the signal line. This controller needs extremely little space.

Profi Control KUAX 680S

The KUAX 680S is a modularly constructed input and putput device. For real-life applications it is equipped with modules for reading digital inputs or setting digital outputs. It needs extremely little space and has a PROFIBUS interface. It can thus network with masters such as the KUAX 680I or the KUAX 657P. Kuhnke deliver it equipped with 4 or 8 module slots.

Profi Control KUAX 681M

The KUAX 681M is a decentralized positioning controller made by Kuhnke. It is a compact device which is attached directly to the motor. It is available for different motors (BG 63 x 55; 24 V and 40 V DC; BG 83 x 90; 40 V DC) and has 2 PROFIBUS interfaces. It can thus network with masters such as the KUAX 680I or the KUAX 657P.

Profile

To be able to network different devices – even of different manufacturers – with PROFIBUS, the PROFIBUS standard must be applied to devices and the various tasks in a clearly defined form. Included in the standard already are basic provisions such as technical interfaces, protocols and services for OSI layers 1, 2 and 7. In addition to this basic standard, further characteristics and dynamic functions of the devices must be standardized for specific applications required by the various fields of industry. Selecting and determining the remaining undefined characteristics (objects, dynamic functions etc.) results in so-called PROFIBUS profiles. Apart from the PROFIBUS User Group (PNO), many experts are also concerned with the drawing up of these profiles.

Among other things, profiles allow the classification of devices of the same type on the basis of well-defined characteristics. Kuhnke deliver devices of Controller Profile Class 2 and Sensor/Actuator Class 2.

Project planning

In automation technology, project planning comprises the preparation, setting up and adjustment of automation devices and systems for and to the various fields of application and specific application processes.

Under VEBES, project planning simply means creating a PROFIBUS network. This process can be divided into the following ten different steps:

- starting VEBES
- opening/creating a network
- defining the bus parameters (optional)
- defining the device types (optional)
- defining the bus stations
- defining the connections
- defining block transfer objects and operations (optional)

- saving the network (copying it to disk)
- documenting the network (printout)
- changing to KUBES

Protocol

A protocol is a collection of all the rules and formats needed for data exchange between the same ISO/OSI communication model layers in different devices.

Protocol Data Unit (PDU)

Data exchange between two devices is exclusively carried out on the same layer (same hierarchy level) of the ISO/OSI model. The information to be exchanged is packed into message packages which are called protocol data units. Such a unit is made up of two parts: the protocol control information (PCI), which serves to co-ordinate the protocol processes, and the service data unit (SDU), which contains the user data and information about the used service.

Service

Services are operations aimed at objects. The PROFIBUS communication services can be divided into application services, administration services and network management services. Services of the first group include access operations to the communication objects of an application process such as “read” and “write”.

The administration services only affect objects or communication relationships of a virtual field device, e.g. the reading of an entry in the object dictionary.

Network management services are concerned with context, configuration and error management.

The following is a list of valid services of Kuhnke masters:
(see table overleaf)

Service	Description
Initiate	initiate connection
Abort	abort connection
Reject	reject illegal service request
Status	read device/user status
Identify	read device information (manufacturer, type, release)
Get_OD	read object dictionary
Read	read values of objects
Write	write values of objects
Event-Notification	transmit event notification
Acknowledge-Event-Notification	acknowledge event notification
Alter-Event-Condition-Monitor	enable/disable event

Service Access Point (SAP)

A SAP is the logical interface between a communication channel and the port. Via the SAP, the protocol data units are exchanged between service requester and service provider. Higher layers of the communication hierarchy can use SAPs to access the services of the layer directly below. SAPs contain the range of valid messages and message forms. When a message reaches a SAP, it is checked for its conformity with the set conditions. If so, the message is passed on; if not, the system generates an error message.

Service access points are assigned numbers. Valid numbers are 1-63. However, SAP1 is reserved for management services and SAP63 for broadcast messages. NIL-SAP receives all frames without a SAP number.

SINEC L2-DP

see 'PROFIBUS protocols'

Slave

A slave is the passive partner of a communication relationship. It provides services requested by the master, thus having the function of a server. In bus systems, slaves can only transmit messages after a corresponding request by a master. Slaves in PROFIBUS networks are normally simple devices such as input and output devices, drives etc.

Slave initiative

A slave normally has no access right to the bus and can only transmit messages after the corresponding request of a master. However, in PROFIBUS networks, there are also slaves with initiative. Although they cannot receive the token either to access the bus, they can be polled by a master and respond by transmitting autonomous unconfirmed messages. Thus a slave with initiative can transmit messages, e.g. alarms, which normally require the bus access rights and the corresponding token management functionality.

Slot time

see 'Bus parameters'

Station Delay Responder

see 'Bus parameters'

Target Rotation Time

see 'Bus parameters'

Token

A token is a certain bit sequence which is used to regulate the bus access rights to network systems according to a set of priorities.

Valve Island 799 PROFIBUS

Valve island for 5/2 directional control valves, available from Kuhnke in various designs; the standard configuration has 8 monostable or 8 bistable valves and 0 or 16 sensor inputs. The valve island can be connected to PROFIBUS so that valve switching operations can be controlled by a superset control device. The signals of the sensor inputs are fed back to this controller to report, for example, the reaching of a limit position.

VEBES

VEBES is the network user and configuration software for PROFIBUS network project management. It is a standalone program but it works in close interaction with KUBES (programming software for writing and testing PLC programs). While VEBES is used for network setup and administration purposes, the programs for the individual controllers are written under KUBES.

VEBES functions are responsible for:

- defining communication partners (the bus stations);
- defining the communication relationships between the individual stations;
- defining the bus parameters;
- defining the type of communication (process chart, block transfer; PROFIBUS protocol);
- defining the parameters of new devices.

VFD support services

The VFD (virtual field device, q.v.) contains all objects and object descriptions listed in the object dictionary of a device that can be addressed and used by an application process. VFD support transmits device-specific status information.

VFD support includes the following services:

- status (information about operation state, type of communication, status of the application process);
- unsolicited status (unrequested transmission of the device status without confirmation);
- identify (device information such as type and manufacturer).

Virtual Field Device (VFD)

The term “Virtual Field Device” (VFD) is an abstract model describing a standardized view (i.e. object dictionary open for all stations, uniform interfaces and services) of the real field devices which communicate via the bus. PROFIBUS exclusively works with the Virtual Field Device; mapping VFD information to the real field devices, and vice versa, is carried out via the Application Layer Interface. The VFD is defined by a model behaviour defined by the FMS (Fieldbus Message Specification). There is normally only one VFD to represent a real device although it is possible to have several.

Status and identification services to transmit status information and VFD environment of a device valid for a specific communication relationship are provided by VFD support.

VFD support includes the following services:

- status
- unsolicited status
- identity

C. List of manufacturers of PROFIBUS accessories

Unless otherwise stated, the order numbers given below are Kuhnke order numbers.

Kuhnke PROFIBUS devices

	Order no.
PC Control KUAX 644	644.425.01
Modu Control KUAX 657P	657.425.20 (central processing unit)*
PROFIBUS-Modul KUAX 657P	657.440.20
Profi Control KUAX 680I	680.420.04; 680.420.08; 680.400.04; 680.400.08*
Profi Control KUAX 680S (10 DIP switches)	680.301.04/08
Profi Control KUAX 681M	681.000.00; 681.001.00*
Valve Island 799 FB2 (PROFIBUS)	99.500.03; 99.500.07; 99.500.04; 99.500.08*

* more than one order number as device is available in various types or module configurations; please ask for our catalogue

PROFIBUS accessories

PROFIBUS connector (SUB-D, 9pin; KUAX 644, PROFIBUS module of the KUAX 657P, KUAX 680S)

Set of bus cable connectors (1 x male, 1 x female)	Order no. 680.180.03
---	----------------------

Cable connector, 4pin (KUAX 681M)	Order no. 681.180.08
-----------------------------------	----------------------

alternatively: miniature round plug-type connector, 07 series, type RSMC 4; Karl Lumberg GmbH & Co., Postfach 1360, D-58579 Schalksmühle

Cable connector, 6pin (Valve Island 799)

Male	Order no. 99.180.01
Female	Order no. 681.180.05

Appendix

Set of bus termination connectors (1 x male, 1 x female)	Order no. 680.180.07
Active bus termination connector with independent power supply	Order no. 680.180.10

Plug-type connector for use with KUAX 681M
Cable connector, 4pin Order no. 681.180.08

Bus cables

Dätwyler System und Netzwerk GmbH, Gottfried von Cramm
Str.1, D-85375 Neufahrn
Type Alfaskop S 41 Art. no. 141980

or:
LAPP GmbH Kabelwerk, Schulte-Delitzsch-Str. 25, D-70565
Stuttgart
Type Unitronic -Bus® LD Art. no. 2170203
(cores 1 x 2 x 0.22 mm²)

or:
Siemens AG, Bereich Automatisierungstechnik (Automation
Technology Division), Gleiwitzer Str. 555, D-90475 Nürnberg-
Moorenbrunn

Type SINEC L2 Busleitung (bus cable)	Order no. 6XV1 830-0AH10
Type SINEC L2 Erdverlegungskabel (underground cable)	6XV1 830-3AH10
Type SINEC L2 Schleppkabel (trailing cable)	6XV1 830-3BH10

Repeaters in accordance with the PROFIBUS specification:

Wieland Elektrische Industrie GmbH, Brennerstr. 10-14, D-96045 Bamberg

Power Repeater (for data transfer lines exposed to particularly much interference)	Order no. 83.010.0100.0
--	-------------------------

Order no. 83.010.0100.0

(You need a minimum of 2 repeaters for PROFIBUS operation)

RS485 repeater

Order no. 83.010.0101.0

PROFIBUS interface card for PCs:

Technologie Management Gruppe i-tec GmbH, Vincenz-
Prießnitz-Str. 1, D-76131 Karlsruhe

Type i-tec

PROFIBUS programming card Order no. PB-SMD-MS-H4

KUHNKE software

KUBES 4 D
G

Order no. 680.502.00

Order no. 680.502.11

VEBES D
 GB

Order no. 680.500.00

Order no. 680.500.11

Appendix

Herr Schiewer:

*Bitte den untenstehende Text auf den
Briefumschlag für die Diskette drucken!*

Exercise program disk

Please read the PROFIBUS.WRI file first.

Index

A

access type 4-15
 addressing
 by bit 4-29
 by byte 4-29
 by word 4-30
 external operands 4-27
 ALI 4-63, 4-69, A-1
 analog signals 4-8
 Application Layer Interface. *see* ALI
 application process A-1

B

basic address 3-43
 baud rate 3-10, 4-4
 bi-directional line amplifier 3-9
 BIBS 4-1
 bit time 4-5, A-1
 block transfer 4-2, 4-8, 4-14, 4-18, A-1
 acknowledged 4-37
 between more than two partners 4-38
 master-master 4-37
 master-slave 4-37
 on command 4-49
 block transfer connection 4-35
 block transfer objects 4-13
 broadcast message A-2
 bus 3-4, 3-8, 3-27, A-2
 access protection A-2
 access right A-2
 number of stations 3-9
 parameters A-3

bus control
 at startup 4-69
 during operation 4-69
 bus failure 3-32, 3-38
 bus monitor 3-2
 bus operation 4-69
 monitored by user program 4-69
 bus parameters 4-1, 4-4, 4-6, 4-13
 bus termination 3-6, 3-10, 3-16

C

cell level 1-2
 channel number 4-28
 CI cyclic control interval A-3
 client 3-3
 coded description
 of external operands 4-28
 communication
 between masters
 frames 4-9
 block transfer 4-2, 4-8, A-1
 hierarchy
 cell level 1-4
 field level 1-4
 ISO/OSI reference model 1-4
 levels 1-2
 master-master 4-37
 master-slave 4-37
 OD 4-3
 process chart 4-2, 4-8
 requirements 4-3
 SAP 4-3
 type of 4-2
 communication endpoint A-3

Index

- communication model
 - Application Layer Interface (ALI) A-1
 - broadcast message A-2
 - communication endpoint A-3
 - communication reference (CR) A-4
 - communication relationship A-4
 - communication relationship list (CRL) A-4
 - connection A-5
 - context management services A-6
 - data transfer services A-7
 - Fieldbus Message Specification A-8
 - ISO/OSI communication model A-8
 - multicast message A-11
 - object A-11
 - object dictionary (OD) A-12
 - protocol A-17
 - Protocol Data Unit (PDU) A-17
 - service A-17
 - Service Access Point (SAP) A-18
 - slot time A-19
 - Station Delay Responder A-19
 - Target Rotation Time A-19
 - token A-19
 - Virtual Field Device A-21
 - communication objects
 - data types A-7
 - communication protocol 1-3
 - communication reference (CR) A-4
 - communication relationship A-4
 - communication relationship list 4-8, 4-11
 - communication relationship list (CRL) A-4
 - communication relationships 4-1, 4-3
 - conflict of addresses 3-43
 - connecting junctions 3-6
 - connection A-5
 - attribute A-5
 - types A-6
 - connection attribute 4-16
 - connection type 4-16
 - MMAC 4-16
 - MSAC 4-16
 - MSAC_SI 4-16
 - context management services A-6
 - controller
 - online PC 4-59
 - online PROFIBUS 4-60
 - online V.24 4-59
 - program transfer from PC 4-59
 - RESET mode 4-67
 - CR A-4
 - CRL A-4. *see* communication relationship list
 - cyclic control interval 4-17
- ## D
- data block 4-12
 - data transfer services A-7
 - data type 4-15
 - integer 4-15
 - octet string 4-15
 - unsigned 4-15
 - data types A-7
 - decentralisation 1-5
 - Decentralized Periphery 1-6
 - device
 - define 4-19
 - device type 4-7, 4-13
 - digital signals 4-8
 - display address range 2-7, 4-61
 - display single address 2-7, 4-61
 - dual-port RAM 3-19

- dynamic display 4-61
 - display address range 2-7
 - display single address 2-7
 - module editor 2-7
 - of operands 4-61

E

- EMC - electromagnetic compatibility 3-18
- encoding switch
 - read 3-23, 3-27, 3-29, 3-35, 3-37
 - set 3-28, 3-30
- error codes 4-64
- error messages 4-26
- event notifications 4-26, 4-67
 - of Kuhnke devices 4-68
- exercise
 - block transfer on command 4-49
 - bus control in the master 4-71
 - cyclic block transfer 4-41
- external operands 4-25

F

- failure messages 4-26
- field device 1-2, 1-4
- field level 1-2, 1-3
- Field Message Specification 1-6
- fieldbus
 - open fieldbus system 1-3
- Fieldbus Message Specification (FMS) A-8
- FMS A-8
- FMS protocol A-8

G

- GAP factor 4-4, 4-5
- GAP Update factor A-8
- GrafKEd 4-1
- grounding/EMC 3-18

- group address 4-28

H

- hardware 3-1

I

- initialization A-8
- initiator 4-16
- installation
 - to be observed 2-3
- ISO/OSI reference model 1-4, A-8

K

- KUAX 644 3-20
 - programming 3-39
 - transfer speed 3-21
- KUAX 657P 3-24
 - programming 3-39
- KUAX 680I 3-27
 - programming 3-39
- KUAX 680S 3-30
 - programming 3-39
- KUAX 681M Profi Control 3-33
- KUBES 4-1, A-9
 - driver 3-43
 - module libraries 4-1
 - test functions 4-60, 4-61
- KUBES module
 - parameters
 - own block number 4-38
 - PB_REC 4-38
 - PB_SEND 4-38
 - PB_STAT 4-38
- KUBES modules A-9
 - parameters 4-38
 - partner block no. 4-39
 - partner station address 4-38
 - status 4-39
 - task number 4-39

Index

PB_REC 4-10
PB_SEND 4-10
PB_STAT 4-10

L

level

cell level 1-2
field level 1-2
levels of communication 1-2
management level 1-2
process control level 1-2
levels of communication 1-2
library management program 4-1
line termination resistor 3-6
logic diagram 4-62
trigger 4-62

M

maintenance

to be observed 2-4

management level 1-2

manual

objectives 1-7

use of 1-8

master 3-31, 4-6, A-10

master-master connection 4-10

Modu Control KUAX 657P A-10

mono-master system 1-6

multi-master system 1-6

multicast message A-11

N

network

bus parameters 4-4

create 4-3

network project A-11

network projects 4-1, 4-25

network topology 3-9

O

object 4-15, A-11

object code A-11

object dictionary (OD) 4-3, 4-11,
4-13, 4-14, 4-15, 4-17, A-12

object dictionary index 4-14

object status 4-65

object type 4-15, A-12

array 4-15

event 4-15

simple variable 4-15

OD 4-3, A-12

OD index 4-11, 4-12, 4-14,
4-17, 4-18

OD management services A-13

open fieldbus system 1-3

operand list 4-25

P

partner status 4-66

PB-BSP-4 4-49

PB_BSP_2 4-71

PB_BSP_3 4-41

PB_REC 4-38

PB_SEND 4-38

PB_STAT 4-38

PC Control KUAX 644 A-13

PC plug-in board

installation

CONFIG.SYS 3-19

WIN.INI 3-19

PDU 4-17, A-17

process chart 4-2, 4-8, 4-9,
4-11, 4-14, 4-17, A-13

process status memory 4-8

process chart objects 4-13

process control level 1-2

Profi Control KUAX 680I A-15

- Profi Control KUAX 680S A-15
- Profi Control KUAX 681M A-15
- PROFIBUS 1-3, A-13
 - accessories C-1
 - assembly 3-10
 - bus A-2
 - bus monitor 3-2
 - bus parameters 4-1
 - bus termination 3-6, 3-16
 - cable 3-2, 3-4
 - communication model
 - Application Layer Interface (ALI) A-1
 - communication relationships 4-1
 - connector 3-5
 - 4pin round connector 3-5
 - 6in round connector 3-5
 - 9-pin D-SUB connector 3-11
 - 9pin D-SUB connector 3-5
 - devices made by Kuhnke 3-2
 - equipment needed for
 - installation 3-2
 - error messages 4-63
 - event notifications 4-67
 - interface 3-2, 3-5, 4-60
 - RS-485 3-10
 - RS232 3-20, 3-27
 - RS485 3-5, 3-15, 3-20, 3-27
 - V.24 3-27
 - line address 4-28
 - messages 4-26, 4-62
 - network 4-3, 4-60
 - network configurator software 4-1
 - PC interface 3-8
 - PROFIBUS interface
 - KUAX 644 3-20
 - KUAX 680I 3-27
 - KUAX 680S 3-30
 - KUAX 681M 3-33
 - PC plug-in module 3-2, 3-8, 3-19
 - programming 3-19
 - Valve Island 799 3-36
- PROFIBUS station address
 - KUAX 644 3-22
- profile A-16
- protocol
 - DP 4-2, 4-10, 4-13
 - FMS 4-2, 4-10, 4-13
 - FMS protocol A-8
 - PROFIBUS-DP 3-30
 - PROFIBUS-FMS 3-30
 - SINEC L2-DP A-19
- repeater 3-2, 3-8
- response time B-1
- software by Kuhnke
 - BIBS 4-1
 - GrafKEd 4-1
 - KUBES 4-1
 - VEBES 4-1
- spur line 3-6
- station address 3-10, 4-28
 - KUAX 681M 3-34
 - Valve Island 3-37
- status messages 4-66
- terminating station 3-7
- time behaviour B-1
- topology 3-9
- transfer speed
 - KUAX 681M 3-34
 - Valve Island 799 3-38
- PROFIBUS device
 - by Kuhnke
 - KUAX 644 PC Control 3-2
 - KUAX 657P Modu Control 3-2
 - KUAX 680I Profi Control 3-2
 - KUAX 680S Profi Control 3-2
 - KUAX 681M Profi Control 3-2
 - Valve Island 799 3-2
- PROFIBUS protocols A-14

Index

PROFIBUS-DP.

see PROFIBUS-Protokoll

PROFIBUS-FMS.

see PROFIBUS-Protokoll

profile A-16

profile class 1-3, 3-2

program

design 2-6

programming

PC bus 3-41

PROFIBUS 3-41

V.24 interface 3-39

via the PROFIBUS interface 3-19

programming interfaces 3-39

programming network projects with

KUBES 4-25

project planning 3-1, A-16

to be observed 2-3

protocol A-17

FMS 1-6

L2-DP 1-6

Protocol Data Unit (PDU) 4-17,

A-17

pull-down resistor 3-6

pull-up resistor 3-6

R

reliability 2-1

repeater 3-2, 3-8, 3-9, 3-10

regenerating repeater 3-8

RESET 4-67

responder 4-16

response time

setup of measuring instruments

B-1

S

safety 2-1

SAP 4-3, 4-18, 4-35, A-18

server 3-3

service A-17

Service Access Point (SAP) 4-3,

4-13, 4-16, A-18

servicing

to be observed 2-4

SINEC L2-DP 4-13, A-19

slave 3-31, 4-6, A-19

loss of status information of

outputs 4-9

slave initiative A-19

Slot Time 4-4, 4-5, A-19

spur line 3-6

Start <RUN> 2-7

station 4-6

address 3-10, 4-6, 4-28

programmable 4-25

symbolic names 4-6

Station Delay Responder 4-4,

4-5, A-19

stations

device type 4-7

status messages 4-26, 4-35

PB_REC 4-40

PB_SEND 4-39

PB_STAT 4-40

Stop 2-7

Symbol Table 4-26

T

T-junctions 3-6

target group 2-1

Target Rotation Time 4-4, 4-5,

A-19

time behaviour B-1

time parameters 4-4

Gap Factor 4-5

Slot Time 4-4, 4-5

Station Delay Responder 4-4, 4-5

Target Rotation Time 4-4, 4-5

- timer parameters
 - GAP factor 4-4
- token 4-5, A-19
- topology 3-9
 - line 3-9
 - star structure 3-9
 - tree structure 3-9
- transfer speed 3-32
- transmission rate 3-4, 3-10

U

- user information
 - other than manual 1-9

V

- Valve Island 799 3-36
 - programming 3-39
- Valve Island 799 PROFIBUS A-20
- VEBES 4-1, 4-25, A-20
 - create network 4-3
 - station selection list 4-3
- VFD A-20, A-21
- VFD support services A-20
- Virtual Field Device A-21

W

- wait loop 4-36
- WIN.INI 3-41

