Kuhnke Electronics
Instruction Manual

CanControl 691

PLC with IP65 Protection

E 623 GB 25.04.2002 / 86.344

This instruction manual is primarily intended for use by design, project, and development engineers. 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 - on whatever legal grounds - are excluded except for instances of deliberate intent or gross negligence on our part.

We reserve the rights for errors, omissions and modifications.

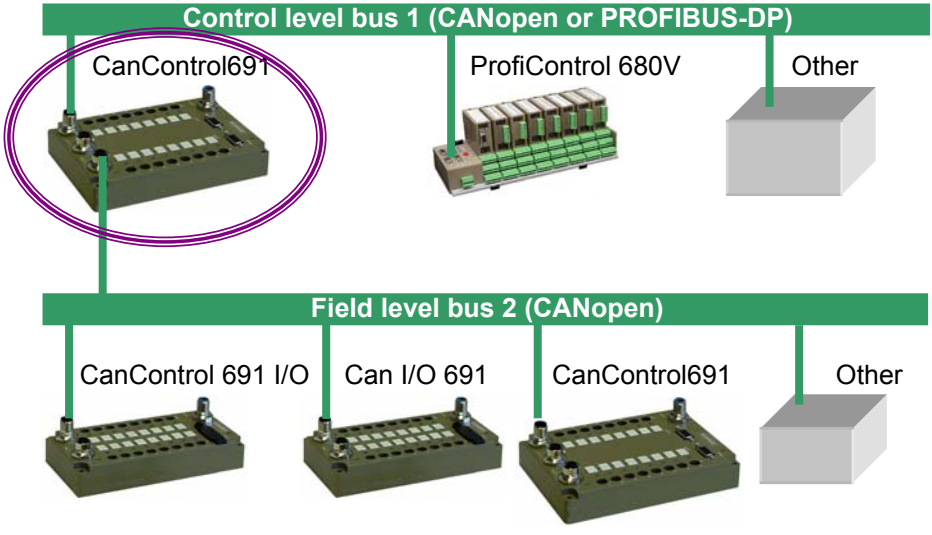
Reproduction even of extracts only with the editor's express and written prior consent.

Table of Contents

1 Introduction

CanControl 691 is a powerful PLC in a compact housing. Its IP 65 protection allows you to install it immediately at or in the machine. The remote CanControl 691 I/O module connect to the bus ports.
The advantage of this approach is that you can do without further housings or switching cabinets and that the distance between sensors/actuators and modules is reduced to the absolute minimum.
The unit features two bus ports, **Bus 1** (CANopen or PROFIBUS-DP) and **Bus 2** which make it the perfect junction box for two networks, e.g. one control-level network and a field-level network close to the actual process.

Example:



The CANopen and PROFIBUS-DP field bus architectures support networks of devices made by different manufacturers. You can therefore use CanControl 691 in any application without being constrained to use but systems supplied by Kuhnke.

1.1 Variants

There are two CanControl 691 variants which differ in their bus connection:

- **CanControl 691-C**
  Bus 1: CANopen
  Bus 2: CANopen
- **CanControl 691-DP**
  Bus 1: PROFIBUS-DP (DP slave)
  Bus 2: CANopen

Terminology

The following convention applies throughout this manual:

- **CanControl 691**
  will be used for information relevant to all variants
- **CanControl 691-C**
  will be used for information relevant to units with two CANopen ports only
- **CanControl 691-DP**
  will be used for information relevant to units with PROFIBUS-DP connected to Bus 1

1.2 From Stand-alone Controllers to Distributed Networks

Remote, intelligent units such as programmable logic controllers (PLCs) play an important role in industrial automation. There are three main reasons for this:

- they are universally applicable,
- programming is easy and comprehensible,
- they provide numerous means of testing and start-up.

As a problem-orientated micro-computer, PLCs have taken over more and more elements of process computing systems in accordance with their permanently growing capacities. They have become universal instruments of automation which have found acceptance in a wide range of action.

A strong tendency towards hierarchical process control systems has developed since. These are marked by task separation. Each part system executes tasks according to its optimal aptitude.

PLCs usually act at the process interface level whereas PCs are used for computations and the management of large data volumes at the control level.

1.2.1 Task Separation Leads to Decentralisation

By adding further components such as sensors and actuators we get to network systems at the field level.

High-capacity interfaces and transmission lines are of greatest importance via which PLCs communicate with further PLCs, other devices and with PCs.

1.2.2 Advantages of Decentralisation

- Reduction of multi-core cables,
- Material (cables, connectors, ...)
- Space (conduits, terminal blocks, switching cabinet)
- Installation (time, possibilities of errors)
- Increased functionality
- The program structure is similar to the object structure, this leading to more transparency
- Reaction to problems (if one part of the system fails other parts can continue to work)
- Reduced setup times
- Pre-testing of individual stations
- Devices of different manufacturers can be combined

These advantages can only be fully effective if a standardised solution for all part-systems is available. In this context, speed and reliability of data transfer and the design as an open system are of decisive importance.

The networkability of the CanControl 691 family ensures that this system fully satisfies the above demands.

2 Reliability, Safety

2.1 Target Group

This instruction manual contains all information necessary for the use of the described product (control device, control terminal, 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 agreements with our suppliers,
- measures for the prevention of static charge during the handling of MOS circuits,
- worst case planning and design of all circuits,
- inspections at various stages of fabrication,
- computer-aided tests of all assembly groups and their interaction in the circuit,
- statistical assessment of the quality of fabrication and of all returned goods for the immediate taking of appropriate corrective actions.

2.3 Notes

Despite the measures described in chapter 2.2, the occurrence of faults or errors in electronic control units - even if most highly improbable - must be taken into consideration.

Please pay particular attention to the additional notes which we have marked by symbols in this instruction manual. While some of these notes make you aware of possible dangers, others are intended as a means of orientation. They are described further down below in descending order of importance.

2.3.1 Danger

This symbol warns you of dangers which may cause death or grievous bodily harm if operators fail to implement the precautions described.

2.3.2 Attention

This symbol draws your attention to information you must take a look at to avoid malfunctions, possible material damage or even dangerous states.

2.3.3 Note

This symbol draws your attention to additional information concerning the use of the described product. It may also indicate a cross reference to information to be found elsewhere (e.g. in other manuals).

2.3.4 Under Construction

This symbol tells you that the function described was not or not fully available at the time this document went to press.

2.3.5 Instruction

You will find a list of instructions wherever you see this symbol in the margin.

It is intended as a means of orientation at places where steps of procedures and background information alternate (e.g. in beginner's manuals).

2.4 Safety

Our products normally become part of larger systems or installations. The information below is intended to help you integrate the product into its environment without dangers to humans or material/equipment.

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

2.4.1 Project Planning and Installation

- 24V DC power supply: Generate as electrically safely separated low voltage. Suitable devices are, for example, split transformers constructed in compliance with European standard EN 60742 (corresponds to VDE 0551).
- In case of power breakdowns or power fades: the program structure is to ensure that a defined state at restart excludes all dangerous states.
- Emergency switch-off installations must comply with EN 60204/IEC 204 (VDE 0113). They must be effective at any time.
- Safety and precautions regulations for qualified applications have to be complied with.
- Please pay particular attention to the notes of warning which, at relevant places, will make you aware of possible sources of dangerous mistakes or faults.
- Relevant standards and VDE regulations are to be complied with in every case.
- Control element installation is to exclude any unintended operation.
- Control cables are to be laid in such a way as to exclude interference (inductive or capacitive) which could influence controller operation or its functionality.

2.4.2 Maintenance and Servicing

- Precautions regulation VBG 4.0 must be observed when measuring or checking a controller in a power-up condition. This applies to section 8 (Admissible deviations when working on parts) in particular.
- Repairs must be carried out by specially trained Kuhnke staff only (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 must be used in modular controllers.
- Modular systems: Always plug or unplug modules in a power-down state. You might otherwise damage the modules or (possibly not immediately recognisably!) inhibit their functionality.
- Always dispose of any batteries and accumulators as hazardous waste.

2.5 Electromagnetic Compatibility

2.5.1 Definition

Electromagnetic compatibility is the ability of a device to function satisfactorily in its electromagnetic environment without itself causing any electromagnetic interference that would be intolerable to other devices in this environment.

Of all known phenomena of electromagnetic noise, only a certain range occurs at the location of a given device. This noise depends on the exact location. It is defined in the relevant product standards.

The international standard regulating construction and degree of noise resistance of programmable logic controllers is IEC 1131-2 which, in Europe, has been the basis for European standard EN 61131-2.

2.5.2 Resistance to Interference

- Electrostatic discharge, ESD
  in acc. with EN 61000-4-2, 3rd degree of sharpness
- Irradiation resistance of the device, HF
  in acc. with EN 61000-4-3, 3rd degree of sharpness
- Fast transient interference, burst
  in acc. with EN 61000-4-4, 3rd degree of sharpness
- Immunity to damped oscillations
  in acc. with EN 61000-4-12 (1 MHz, 1 kV)

2.5.3 Interference Emission

Interfering emission of electromagnetic fields, HF
in acc with EN 55011, limiting value class A, group 1

If the controller is designed for use in residential areas, then high-frequency emissions must comply with limiting value class B as described in EN 55011.

You may wish to install the controller in an earthed metal cabinet and/or add filters to the supply cables.

2.5.4 General Notes on Installation

As component parts of machines, facilities and systems, electronic control systems must comply with valid rules and regulations, depending on their field of application.

General requirements concerning the electrical equipment of machines and aiming at the safety of these machines are contained in Part 1 of European standard EN 60204 (corresponds to VDE 0113).

For safe installation of our control system please observe the following notes (→ 2.5.5 and following).

:

2.5.5 Protection against External Electrical Influences

Connect the control system to the protective earth conductor to eliminate electromagnetic interference. Ensure practical wiring and laying of cables.

2.5.6 Cable Routing and Wiring

Lay power supply circuits separately, never together with control current loops:

- DC voltage 60 V ... 400 V
- AC voltage 25 V ... 400 V

Joint laying of control current loops is allowed for:

- shielded data signals
- shielded analogue signals
- unshielded digital I/O lines
- unshielded DC voltages < 60 V
- unshielded AC voltages < 25 V

2.5.7 Location of Installation

Make sure that there are no impediments due to temperatures, dirt, impact, vibration and electromagnetic interference.

2.5.7.1 Temperature

Consider heat sources such as general heating of rooms, sunlight, heat accumulation in assembly rooms or control cabinets.

2.5.7.2 Dirt

IP 65 protection means that the unit is dust and jet water tight.

2.5.7.3 Impact and Vibration

Consider possible influences caused by motors, compressors, transfer lines, presses, ramming machines and vehicles.

2.5.7.4 Electromagnetic Interference

Consider electromagnetic interference from various sources near the location of installation: motors, switching devices, switching thyristors, radio-controlled devices, welding equipment, arcing, switched-mode power supplies, converters / inverters.

2.5.8 Particular Sources of Interference

2.5.8.1 Inductive Actuators

Switching off inductances (such as from relay coils, contactors, solenoid valves or actuating magnets) produces voltage peaks. It is necessary to reduce these extra voltages to a minimum.
Reducing elements may be diodes, Z-diodes, varistors or RC elements. To find the best adapted elements, we recommend that you contact the manufacturer or supplier of the corresponding actuators for the relevant information.

3 Hardware

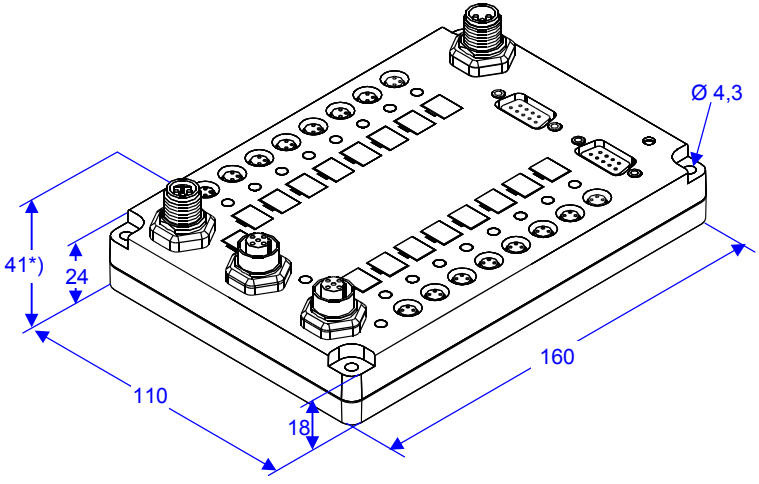
3.1 Mechanical Design

The unit consists of plastic housing with IP 65 protection.

The protection only works if all connectors are covered (either by the mating connector or by a dummy plug → 8.2.3 Accessories).

3.1.1 Dimensions

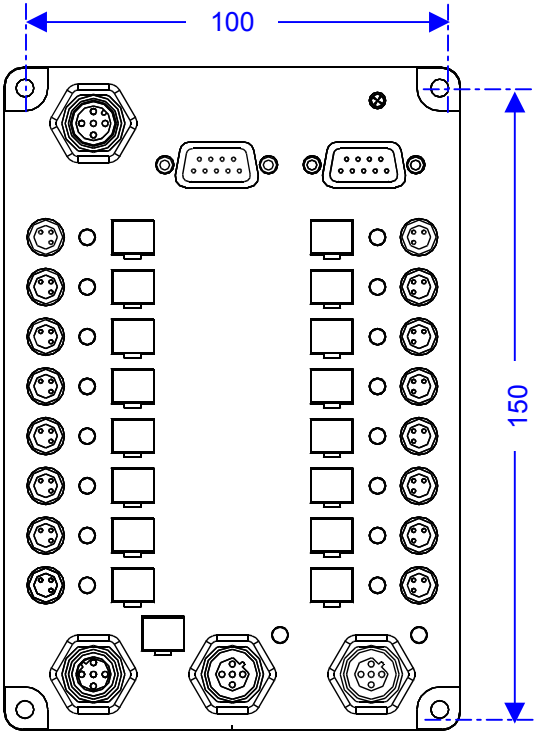
All dimensions [mm]:



**) Please note that the connectors and wires increase the space requirements.*

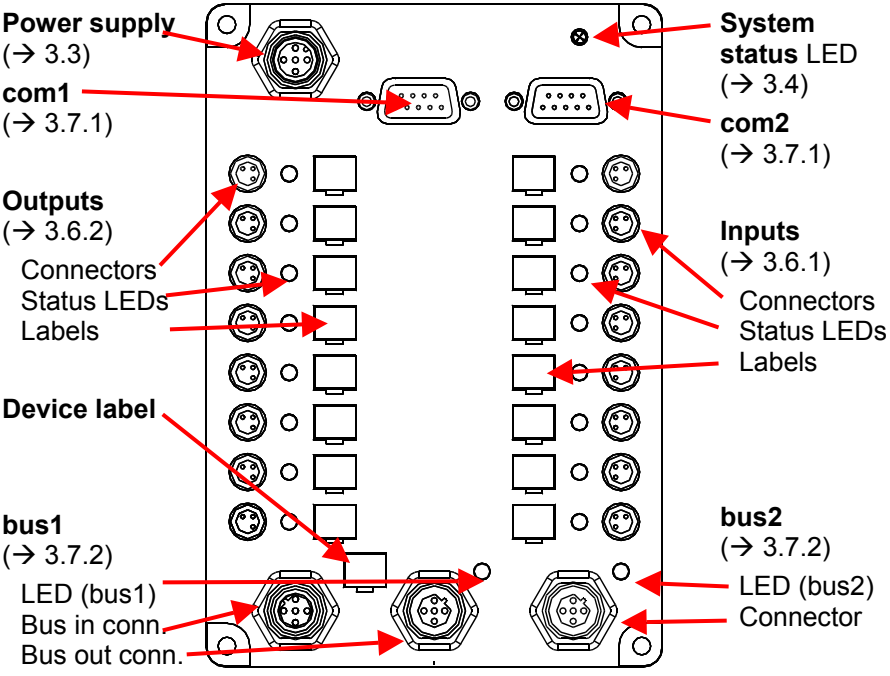
3.1.2 Installation

The unit is designed for screw-fit installation. There are 4 drill holes for the M4 screws.



3.2 Front View

The figure illustrates where the connectors and light emitting diodes (LEDs) are located:



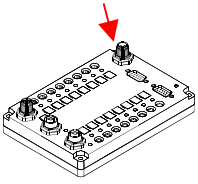
See the next sections for details.

3.3 Power Supply

The power supply connects to the male round 5-pin plug. The connector has a "standard coding".

Pin wiring:

Pin	Function
1	+24 VDC supply to **system and inputs**
2	+24 VDC supply to **outputs**
3	0 V
4	-
5	**-**



Although the outputs are supplied separately, their potential is not separated from the "system and inputs" supply. All voltages reference the shared 0 V connector.

A separate supply allows you to cut the outputs off the supply without resetting the system.

3.3.1 Connectors

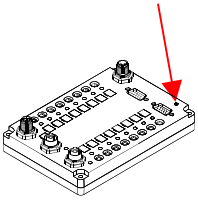
The cable connectors are not delivered with the device. Refer to the appendix (→ 8.2.3.1) to find a list of sources where you can purchase connectors and fully assembled cables.

Colour coding of the WWAK4 cable supplied by Escher

Pin	Function	Escher cable
1	+24 VDC supply to **system and inputs**	Brown
2	+24 VDC supply to **outputs**	White
3	0 V	Blue
4	-	-
5	**-**	

3.4 System Indicators

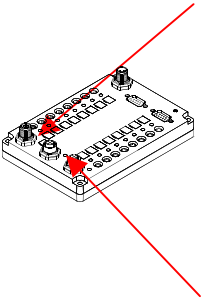
3.4.1 Status LED



The **status** LED indicates the state of the controller:

LED	Function
Off	No power supplied to **system and inputs**
On (green)	Ready
Red (flashing)	Problem (→ 7.2 Status LED)

3.4.2 Bus Communication



➢ **bus1**
The yellow **bus** LED indicates the status of bus communication via bus 1:

LED	Function
Off	No communication No bus configuration
On	Communication OK
Flashing	Warning, no communication (bus configuration exists)

➢ **bus2**
The yellow LED indicates the status of bus communication via bus 2:

LED	Function
Off	No communication No bus configuration
On	Communication OK
Flashing	Warning, no communication (bus configuration exists)

3.5 Memory

There is a lot of memory capacity for storing programs and data.

The memory is divided up differently in devices with an IEC task (for programming using KUBES IEC 61131-3). There were no details available before this manual went to press.

3.5.1 Internal Non-volatile Flash EPROM

- Size: 256 kbyte
- Access time: 55 ns
- Function: program memory

This memory safely hosts the user program, which can be updated via the serial interfaces using the appropriate programming tools.

3.5.2 Internal Volatile RAM

- Size: 240 kbyte
- Access time: 55 ns
- Function: Data memory

This memory stores operational data. It is cleared every time you restart the system; you should therefore avoid storing remanent data in it.

3.5.3 Internal Non-volatile NV-RAM

- Size: 32 kbyte
- Access time: 45 ns
- Function: Data memory

This memory stores operational data. You can also store remanent data in it because a NV-RAM saves data also when it is not supplied with energy, i.e. it is not cleared at restart.

All remanent operands (→ 4.3.2) are deposited in this memory.

8 kbyte of this memory are reserved for storing up to 12 data modules.

For further information on data modules please refer to Instruction Manual E386GB, KUBES Modules (DBCOPY).

3.6 Inputs and Outputs

This unit has 8 digital inputs and 8 digital outputs. Look at figure (→ 3.2) to locate the connectors and relevant status LEDs.

3.6.1 Inputs

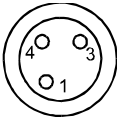
This inputs are the in-gates for input signals, which the user program operands I00.00…07 can read just like the normal inputs. The default input delay is approx. 0.5 ms. Another application of the 8 inputs is the generation of processor interrupts which allow you to let the user program immediately react to the triggering event (→ 4.6.2).

3.6.1.1 Connectors

The cable connectors are not delivered with the device. Refer to the appendix (→ 8.2.3.2) to find a list of sources where you can purchase connectors and fully assembled cables.

Every input has its own connector of type "round 8 mm snap-on, 3-pin, female".

All 3 terminal points have been assigned. This allows you to not only connect (push-button) switches but also sensors that need a 24 VDC supply. The 24 VDC are fed in through supply pin 2 (→ 3.3).



Pin	Signal	Core wire[1]
1	+24 V DC	Brown
3	0 V	Blue
4	24 VDC input signal	Black

- Logical 0: 0 ... 5 V
- Logical 1: 15 ... 28 V

[1] The standard sensor and actuator cables are colour-coded as specified

3.6.1.2 Defined Signals and Switching Thresholds

- Logical 0 ≤ 5 V
- Logical 1 ≥ 15 V (hysteresis 1...4 V

3.6.1.3 Signal Delay

Peak voltages (noise impulses) are filtered out to avoid them being taken as valid signals which might start unintended switching actions. This delays signal detection.

3.6.2 Outputs

Digital outputs make the connection to the external actuators (relays, contactors, solenoids, valves...).

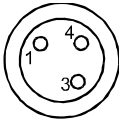
They can switch capacitive and inductive loads. They are equipped with free-wheeling diodes to suppress inductive switch-off peaks. The output status is indicated by LEDs.

You can set the outputs to being actuated either by the user program or by a CAN bus station. Transfer addresses are used to set the access to blocks of 4 outputs which are actuated either by the user program or by bus 1 or by bus 2 (→ 4.6).

3.6.2.1 Connectors

The cable connectors are not delivered with the device. Refer to the appendix (→ 8.2.3.2) to find a list of sources where you can purchase connectors and fully assembled cables.

Only 2 of the pins have been assigned: The 24 VDC for the output signal are fed in through supply pin 1 (→ 3.3).



Pin	Signal	Core wire
1	24 VDC	
3	0 V	Blue
4	24 VDC output signal	Black

3.6.2.2 Short Circuit and Overload Protection

The outputs are protected against destruction due to overload or short circuit:

- The load current is limited to max. 500 mA.

Actuation

- A temperature monitor switches the output off after 0.1 to 1 s and tells the CPU that there is a short circuit.
- The CPU outputs a short-circuit message,
- indicates the short circuit by flash code (1) of LED "status",
- and actuates interrupt module 18 (→ 7.5.1).

Restart

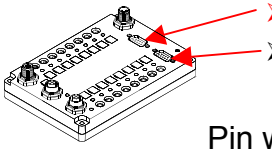
- Find the source of the problem
- Switch off the unit (no voltage)
- Remove the problem
- Supply voltage to the unit again

3.7 Communication Ports

This unit has 4 built-in communication ports. Look at figure (→ 3.2) to locate the connectors and relevant status LEDs.

3.7.1 V.24 Ports

The unit features 2 V.24 ports (RS 232 interfaces). They are located next to the power supply connector.



Pin wiring

The terminal points of the two D-Sub sockets are assigned to the following signals:

Core no.	Signal
1	unused
2	TxD
3	RxD
4	unused
5	Gnd
6	unused
7	CTS
8	RTS
9	unused
Housing	earth

Connect the cable shield to the plug casing.

3.7.1.1 Use of the V.24 Port

The V.24 is mainly used as a programming interface. It is set to the KUBES protocol by default.

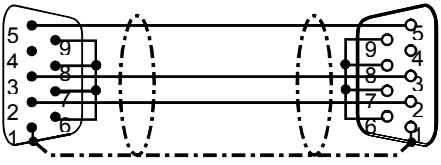
The port can be used for data exchange with dialogue terminals, PCs or other devices, for example.

As there are two V.24 ports, CanControl 691 can communicate with two devices, e.g. programming PC and dialogue terminal, at the same time.

3.7.1.2 Programming Cable

Use the standard programming cable to connect the control unit with the.

Part number	691.151.09
Length	3,0 m
Signal	*Control unit* *PC*



If you need IP 65 protection use a preassembled cable with permanently attached and sealed wires.
The cable may have a 1:1 wiring. The connection with CanControl 691 must be secured by means of the rubber seal that is delivered with the control unit package.

3.7.2 Bus Ports

The unit features two bus ports, **Bus 1** (CANopen or PROFIBUS-DP) and **Bus 2** (CANopen).

3.7.2.1 Connectors

The cable connectors are not delivered with the device. Refer to the appendix (→ 8.2.3.4) to find a list of sources where you can purchase connectors and fully assembled cables.

3.7.2.2 Potential Separation

The potentials of the bus ports are separated. One consequence is that no compensation currents can go through the bus line.

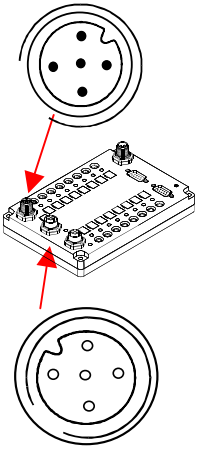
3.7.2.3 Cable Shielding and Earthing

Connect the cable shield to the plug casing. This ensures a conducting connection between the shield and the mounting plate via the metal arrester rail and the screws. Earth the system well to protect it against high-frequency interference by absorbing and arresting the interfering current.

3.7.2.4 Bus 1 Used as a CANopen Port

Name of the variant: **CanControl 691-C.**

There are two bus plugs connected in parallel. You can therefore easily implement a through connection to the next bus station.



➢ **bus1 in** (male)

Pin	Function	
1	Cable shield (capacitive connection to GND)	
2...3	-	
4	CAN_H	CiA Draft Standard 301 Version 3.0
5	CAN_L	

➢ **bus1 out** (female)

Station address

Run the CAN configuration software to set the address.

3.7.2.5 Bus 1 Used as a PROFIBUS-DP Slave Port

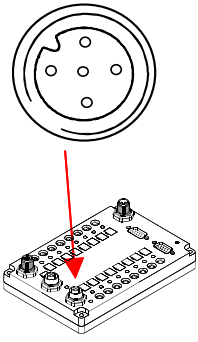
Name of the variant: **CanControl 691-DP**.

This variant was not available yet when the manual went to press.

3.7.3 Bus 2 Connection

Bus 2 is a CANopen port in all variants.

There is one connector that you attach the remote input/output devices to – e.g. CanControl 691 I/O – which are located somewhere in the machine configuration to pick up sensor signals or to output signals to actuator elements. You have the option, of course, to connect other CANopen units to this port.



➢ **bus2** (female)

Pin wiring

Pin	Function	
1...3	-	
4	CAN_H	CiA Draft Standard 301 Version 3.0
5	CAN_L	

Station address

Run the CAN configuration software to set the address.

Bus termination

The controller features a built-in bus termination. You must therefore attach the unit as the first or last station of a bus line. Do not start any branch lines at this point!

4 Functions

4.1 PLC Function

4.1.1 Working Method

The microprocessor for the user program retrieves its instructions from two programs: the operating system and the user program.

4.1.1.1 Operating System

The operating system (monitor) is stored in the internal flash EPROM (→ 3.5.1). It defines all of the controller's properties. The operating system is delivered with the CPU.

4.1.1.2 User Program

The user program contains the instructions required for controlling the machine or system. It is mapped as a module architecture. KUBES is the PC-based programming environment used to write the software modules. KUBES can go through the following port(s) to access the controller (online):

- **com1** and **com2** (→ Communication Ports)

These two V.24 interfaces (RS 232) support the KUBES protocol. To program the controller, attach one end of the programming cable (657.151.03) to the controller's com1 or com2 outlet and the other to a COM port of the programming PC.

The next sections will detail the basics required for writing user programs for CanControl 691.

Refer to the KUBES instruction manual (E 327 GB) to learn how to plan, write and implement user programs. The module structure, commands, operands, etc. are explained in the programming manual (E 417 GB).

4.1.2 I/O Process Map

The controller uses so-called process maps to communicate with the I/O-level devices. The process maps are updated permanently by downloading input information and uploading output information.

4.1.2.1 Internal Process Map

The internal process map contains the data of the I/Os directly connected to the controller. The I/O data map is refreshed at the end of every program cycle.

4.1.2.2 External Process Map (CANopen)

The data of the remote CANopen peripherals is updated by means of a CANopen task whose processing cycle is not synchronised with the user program cycle. External data may therefore change within a program cycle. You must therefore follow a special routine to ensure data consistency of read/write operations (→ instruction manual E 615 GB, CANopen).

4.1.2.3 External Process Map (PROFIBUS)

The data of the remote PROFIBUS peripherals is updated by means of a PROFIBUS task whose processing cycle is not synchronised with the user program cycle. External data may therefore change within a program cycle. You must therefore follow a special routine to ensure data consistency of read/write operations (→ instruction manual E 611 GB, PROFIBUS-DP).

4.2 PLC Status

4.2.1 RUN

The user program is being processed. The central input/output data are being refreshed via the internal process map.

4.2.1.1 CANopen

In a CANopen network, CAN will be put into Operate mode; otherwise it will remain Offline. Data will be transferred via the external process map which is not synchronised with the user program cycle.

4.2.1.2 PROFIBUS

Still at the discussion stage when this manual went to press.

4.2.2 STOP

The user program will no longer be processed. The central input/output data are being refreshed via the internal process map. KUBES (for example) allows you to manipulate the data.

4.2.2.1 CANopen

CANopen will be put into Operate mode. Data transfer will not be synchronised with the user program cycle.

4.2.2.2 PROFIBUS

Still at the discussion stage when this manual went to press.

4.2.3 STOP and Reset

The user program will no longer be processed. The central input/output data will be reset; they will be no longer refreshed via the internal process map. Non-remanent data will also be reset.

4.2.3.1 CANopen

CANopen will be put into Clear mode. Zeros will be sent to the slave outputs. The reaction of the remote data depends on the peripheral (slave).

4.2.3.2 PROFIBUS

Still at the discussion stage when this manual went to press.

4.2.4 Test

Test mode is started via the KUBES programming software. The test mode provides program analysis functions such as slow mode, single step, break point processing with regard to the current I/O and processor states.

4.3 Standard KUBES Programming

This section refers to programming using the standard KUBES features only.

*Install the **KUBES IEC 1131-3** plug-in to extend KUBES' functionality by the programming methods in accordance with IEC 1131-3. Refer to the separate manual (→ references in ch. 8.3) for details.*

4.3.1 Operands

All memory cells that can be addressed by the user program for signal processing or data storage are referred to as operands. They are "operated with". There are two basic types of operands:

- **Local operands**
  They can be immediately addressed by the user program in the control unit as inputs, outputs or memory cells.
- **External operands**
  They are available immediately in the controller's memory. CANopen or PROFIBUS-DP use them as a process map of the other partner stations on the bus.

4.3.2 Local Operands

4.3.2.1 Digital Inputs and Outputs

Group	Input	Function	Type	Quantity	Input Range		Comment
					from	to	
I	Ixx.yy	Inputs	Bit	8	I00.00	I00.07	Local I/Os
O	Oxx.yy	Outputs		8	O00.00	O00.07	

- **Inputs** "read" the signals of switches, push-buttons, initiators, etc., and add the signal status to the process map between PLC cycles.
- **Outputs** upload control signals to relays, contactors, solenoids, etc., to switch them on or off. The CPU writes the signals to the process map as instructed by the user program. Between PLC cycles, the output data in the process map are passed on to the outputs You can use output addresses O00.08…O15.15 (not mentioned in the table above) as internal marker addresses (→ 4.3.2.2).

4.3.2.2 Internal Markers

Group	Input	Function	Type	Quantity	Input Range from	Input Range to	Comment
Bit markers							
M	Mxx.yy	Markers (non-remanent)	Bit	256	M00.00	M15.15	Automatically reset (=0) after restart or reset.
SM				256	SM00.00	SM15.15	
LM	LMxx.yy			256	LM00.00	LM15.15	
FM	FMxx.yy			256	FM00.00	FM15.15	
O	Oxx.yy			248	O00.08	O15.15	Output addresses not supported by the hardware
R	Rxx.yy	Markers (remanent)		256	R00.00	R15.15	Permanently stored in the NV-RAM (→ 3.5.3).
SR	SRxx.yy			256	SR00.00	SR15.15	
Byte markers							
BM	BMxx.yy	Byte marker (non-remanent)	Byte	256	BM00.00	BM15.15	Automatically reset (=0) after restart or reset.
SBM	SBMxx.yy			256	SBM00.00	SBM15.15	
BI	BIxx.yy			256	BI00.00	BI15.15	
BO	BOxx.yy			256	BO00.00	BO15.15	
BR	BRxx.yy	Byte marker (remanent)		256	BR00.00	BR15.15	Permanently stored in the NV-RAM (→ 3.5.3).
SBR	SBRxx.yy			256	SBR00.00	SBR15.15	
ABM	ABMxx.yy			256	ABM00.00	ABM15.15	
BC	BCxx.yy			256	BC00.00	BC15.15	
SBC	SBCxx.yy			256	SBC00.00	SBC15.15	
BD	BDxx.yy			256	BD00.00	BD15.15	
SBD	SBDxx.yy			256	SBD00.00	SBD15.15	
LBM	LBMxx.yy			256	LBM00.00	LBM15.15	
FBM	FBMxx.yy			256	FBM00.00	FBM15.15	

The CPU makes 1656 bit markers in 13 groups and 3328 byte markers in 11 groups available for storing ("marking") current data. 512 of the bit markers and 2304 of the byte markers are remanent markers stored in the NV-RAM (→ 3.5.3).

4.3.2.3 Data Processing Ranges

Group	Input	Function	Type	Quantity	Input Range		Comment
					from	to	
DB0	DB0xx.yy	Data processing ranges for data modules (blocks)	Byte	256	DB000.00	DB015.15	Automatically reset (=0) after restart or reset.
DB1	DB1xx.yy			256	DB100.00	DB115.15	
DB2	DB2xx.yy			256	DB200.00	DB215.15	
DB3	DB3xx.yy			256	DB300.00	DB315.15	
DB4	DB4xx.yy			256	DB400.00	DB415.15	
DB5	DB5xx.yy			256	DB500.00	DB515.15	
DB6	DB6xx.yy			256	DB600.00	DB615.15	
DB7	DB7xx.yy			256	DB700.00	DB715.15	

The data processing ranges are used to add data to data modules (data blocks) or to copy them from there.

They can also be used like normal byte markers (BM, SBM, BO).

4.3.2.4 Transfer Addresses

Transfer addresses (also referred to as dual-port RAM) are the interface between the user program and special functions.

Addresses not supporting any special functions can be used like normal byte markers.

Group	Input	Function	Type	Quantity	Input Range		Comment
					from	to	
SLQ	SLQxx.yy	Transfer addresses (internal I/Os) → 4.6.1	Byte	256	SLQ00.00	SLQ15.15	Automatically reset (=0) after restart or reset.
SLR	SLRxx.yy	Byte marker (non remanent)	Byte	256	SLR00.00	SLR15.15	
SLS	SLSxx.yy		Byte	256	SLS00.00	SLS15.15	
SLT	SLTxx.yy		Byte	256	SLT00.00	SLT15.15	

4.3.2.5 Timers, Counters, Clock Markers

Group	Input	Function	Type	Quantity	Input Range from	Input Range to	Comment
Software timers							
PT	PTxx.yy	Timer	Word	128	PT00.00	PT07.15	Software timer
Software counters							
C	Cxx.yy	Counter	Word	32	C00.00	C01.15	Software counter
Clock markers							
T	Txx.yy	10 ms clock	Byte	1	T00.00	-	Automatically increments the byte in the set clock interval
		100 ms clock		1	T00.01	-	
		1 s clock		1	T00.02	-	
		10 s clock		1	T00.03	-	

Timers

The controller has a default set of 128 software timers (PT00.00-PT07.15). The time range is 10 ms - 65535 s. The timers can be programmed with a rising or falling delay or as clock / pulse generators. You can define them to be remnant, i.e. they will be stored in the NV-RAM (→ 3.5.3).

Counters

You can set up 32 counters as up or down counters with a resolution of 16 bit (0-65535). They can be made remanent, too.

Clock markers

The four clock markers are interrupt-controlled and incremented at the set interval. An overflowing count automatically starts a new cycle (at 0!).

4.3.2.6 Special Operands

Group	Input	Function	Type	Quantity	Input Range		Comment
					from	to	
Programmable pulse							
PP	PPxx.yy	Programmable pulse	Bit	128	PP00.00	PP07.15	Edge detection
Logical value							
PL	PLxx.yy	logical 0	Bit	1	PL00.00	-	Set logical value (0/1)
		logical 1		1	PL00.01	-	
Cycle time count							
SLQ	SLQxx.yy	Current cycle time	Word	1	SLQ02.08	-	Current cycle time [* 100 µs]
SLQ	SLQxx.yy	Maximum cycle time	Word	1	SLQ02.10	-	Maximum cycle time [* 100 µs]
System error marker							
ERR	ERR00.00	System error	Byte	1	ERR00.00	-	(→ 7)

Programmable pulse

Use the 128 pulse flag bits to detect the edge of signal impulses.

Logical value

The two operands are set to exactly one value each.

Cycle time

The cycle time of the user program is monitored internally and shown in two byte operands (16 bit) in transfer memory block SLQ.

- SLQ02.08...09 Current cycle time [* 100 µs]
- SLQ02.10...11 Maximum cycle time [* 100 µs]

System error marker

The monitor program writes any system errors into byte operand ERR00.00 (8 bit) which the user program can read to react accordingly (→ 7).

4.3.3 External Operands

The external operands, too, are permanently stored in the controller's memory. They map the process signals of the external bus stations communicating via the relevant bus.

The external operands listed below cannot be used directly as parameters of KUBES modules and function modules. Instead, you will have to copy them to standard byte markers first.

4.3.3.1 CANopen Operands Bus 1

Applies to variant **CanControl 691-C**.

The operands below are assigned the specified function if and when the unit is connected to a CANopen bus via port Bus 1.

Group	Input	Function	Type	Quantity	Input Range		Comment
					from	to	
CANopen process map							
DNA	DNAxx.yy	CANopen process map of outputs	Byte	256	DNA00.00	DNA15.15	Automatically reset (=0) after restart or reset.
DNB	DNBxx.yy			256	DNB00.00	DNB15.15	
DNC	DNCxx.yy			256	DNC00.00	DNC15.15	
DND	DNDxx.yy			256	DND00.00	DND15.15	
DNE	DNExx.yy	CANopen process map of inputs		256	DNE00.00	DNE15.15	
DNF	DNFxx.yy			256	DNF00.00	DNF15.15	
DNG	DNGxx.yy			256	DNG00.00	DNG15.15	
DNH	DNHxx.yy			256	DNH00.00	DNH15.15	
CANopen messages							
DSA	DSAxx.yy	CANopen status	Byte	256	DSA00.00	DSA15.15	Automatically reset (=0) after restart or reset.
DEA	DEAxx.yy	CANopen emergency		256	DEA00.00	DEA15.15	

If you do not use the above port, the operands can act like local byte markers.

4.3.3.2 PROFIBUS Operands Bus 1

Applies to variant **CanControl 691-DP**.

Still at the discussion stage when this manual went to press.

If you do not use the above port, the PROFIBUS operands, too, can act like local byte markers.

4.3.3.3 CANopen Operands Bus 2

Applies to all variants.

The operands below are assigned the specified function if and when the unit is connected to a CANopen bus via port Bus 2.

Group	Input	Function	Type	Quantity	Input Range		Comment
					from	to	
CANopen process map							
CNA	CNAxx.yy	CANopen process map of outputs	Byte	256	CNA00.00	CNA15.15	Automatically reset (=0) after restart or reset.
CNB	CNBxx.yy			256	CNB00.00	CNB15.15	
CNC	CNCxx.yy			256	CNC00.00	CNC15.15	
CND	CNDxx.yy			256	CND00.00	CND15.15	
CNE	CNExx.yy	CANopen process map of inputs		256	CNE00.00	CNE15.15	
CNF	CNFxx.yy			256	CNF00.00	CNF15.15	
CNG	CNGxx.yy			256	CNG00.00	CNG15.15	
CNH	CNHxx.yy			256	CNH00.00	CNH15.15	
CANopen messages							
CSA	CSAxx.yy	CANopen status	Byte	256	CSA00.00	CSA15.15	Automatically reset (=0) after restart or reset.
CEA	CEAxx.yy	CANopen emergency		256	CEA00.00	CEA15.15	

If you do not use the above port, the operands can act like local byte markers.

4.3.4 KUBES Commands

The tables below list all the commands including the types of addressing supported and the memory capacity and processing time required.

Please note the different types of addressing (→ 4.5.3) because the tables only show a limited selection.

Make sure to always operate with operands of the same size (bit, byte, word). Avoid mixing operand sizes because you might otherwise end up with faulty results.

4.3.4.1 Programming of External Operands

The processing times listed in the tables also refer to the external operands (→ 4.3.3.1 and 4.3.3.3, CANopen, and 4.3.3.2, PROFIBUS) if there is an active bus connection. Please add the times required for signal transfer across the bus and for signal processing by the remote device.

When you switch on a network or a bus station you may have to wait a couple of seconds until all interconnections have been established. During that time, the signal states indicated for external inputs are not reliable. The same applies to the actuation of external outputs.

For further information on programming, use of commands and example programs please refer to the Programming Manual (E 417 GB) and the KUBES online help engine.

4.3.5 Logic Commands

Logic commands control logical operations of operands and assign the results of the operations.

They can be used with bit, byte and word operands.

4.3.5.1 Load Commands

Cmnd	Operand (example)	Byte	Function	V[1]	C[1]	Z[1]
L	Load value (bit or byte)					
	I00.00	4	1-bit address			
	BM00.00	4	8-bit address			
	100	4	8-bit constant			
	I00.00[10]	8	1-bit address with constant offset		n	d
	I00.00[BM01.00]	14	1-bit address with variable offset			
	BM00.00[10]	8	8-bit address with constant offset			
	BM00.00[BM01.00]	14	8-bit address with variable offset			
LN	Load negated value (bit or byte)					
	I00.00	6	1-bit address			
	BM00.00	6	8-bit address			
	I00.00[10]	10	1-bit address with constant offset		n	d
	I00.00 [BM01.00]	16	1-bit address with variable offset			
	BM00.00[10]	10	8-bit address with constant offset			
	BM00.00[BM01.00]	16	8-bit address with variable offset			
LD	Load value (double byte) [1]					
	BM00.00	4	Even address			
	BM00.01	10	Odd address			
	10000	4	Constant		n	d
	BM00.00[10]	16	16-bit address with constant offset			
	BM00.00[BM01.00]	22	16-bit address with variable offset			
LDW	Load value (double word)					
	BM00.00		Even address			
	BM00.01		Odd address			
	10000		Constant			
	BM00.00[10]		32-bit address with constant offset			
	BM00.00[BM01.00]		32-bit address with variable offset			

[1] Manipulation of O**V**erflow, **C**arry and **Z**ero flag: n no change of flag
d defined change of flag
u undefined change of flag

[1] See Swap commands (→ 4.3.5.7)

4.3.5.2 AND Commands

Cmnd	Operand (example)	Byte	Function	C[1]	Z[1]
A	AND operation with value (bit or byte)				
	I00.00	4	1-bit address	0	d
	BM00.00	4	8-bit address		
	100	4	8-bit constant		
	I00.00[10]	10	1-bit address with constant offset		
	I00.00[BM01.00]	16	1-bit address with variable offset		
	BM00.00[10]	10	8-bit address with constant offset		
	BM00.00[BM01.00]	16	8-bit address with variable offset		
AN	AND operation with negated value (bit or byte)				
	I00.00	8	1-bit address	0	d
	BM00.00	8	8-bit address		
	I00.00[10]	12	1-bit address with constant offset		
	I00.00[BM01.00]	18	1-bit address with variable offset		
	BM00.00[10]	12	8-bit address with constant offset		
	BM00.00[BM01.00]	18	8-bit address with variable offset		
AD	AND operation with value (double byte)				
	BM00.00	6	Even address	0	d
	BM00.01	10	Odd address		
	10000	4	Constant		

[1] Manipulation of **C**arry and **Z**ero flag:

- n no change of flag
- 0 flag cleared
- d defined change of flag
- u undefined change of flag

4.3.5.3 OR Commands

Cmnd	Operand (example)	Byte	Function	C[1]	Z[1]
O	OR operation with value (bit or byte)				
	I00.00	4	1-bit address	0	d
	BM00.00	4	8-bit address		
	100	4	8-bit constant		
	I00.00[10]	10	1-bit address with constant offset		
	I00.00[BM01.00]	16	1-bit address with variable offset		
	BM00.00[10]	10	8-bit address with constant offset		
	BM00.00[BM01.00]	16	8-bit address with variable offset		
ON	OR operation with negated value (bit or byte)				
	I00.00	8	1-bit address	0	d
	BM00.00	8	8-bit address		
	I00.00[10]	12	1-bit address with constant offset		
	I00.00[BM01.00]	18	1-bit address with variable offset		
	BM00.00[10]	12	8-bit address with constant offset		
	BM00.00[BM01.00]	18	8-bit address with variable offset		
OD	OR operation with value (double byte)				
	BM00.00	6	Even address	0	d
	BM00.01	10	Odd address		
	10000	4	Constant		

[1] Manipulation of **C**arry and **Z**ero flag:

- n no change of flag
- 0 flag cleared
- d defined change of flag
- u undefined change of flag

4.3.5.4 Exclusive OR Commands

Cmnd	Operand (example)	Byte	Function	C[1]	Z[1]
XO	Exclusive OR operation with value (bit or byte)				
	I00.00	4	1-bit address	n	d
	BM00.00	4	8-bit address	0	
	100	4	8-bit constant	0	
	I00.00[10]	10	1-bit address with constant offset	n	
	I00.00[BM01.00]	16	1-bit address with variable offset	n	
	BM00.00[10]	10	8-bit address with constant offset	0	
	BM00.00[BM01.00]	16	8-bit address with variable offset	0	
XON	Exclusive OR operation with negated value (bit or byte)				
	I00.00	8	1-bit address	n	d
	BM00.00	8	8-bit address	0	
	E00.00[10]	12	1-bit address with constant offset	n	
	I00.00[BM01.00]	18	1-bit address with variable offset	n	
	BM00.00[10]	12	8-bit address with constant offset	0	
	BM00.00[BM01.00]	18	8-bit address with variable offset	0	

[1] Manipulation of **C**arry and **Z**ero flag:

- n no change of flag
- 0 flag cleared
- d defined change of flag
- u undefined change of flag

4.3.5.5 Assignment

Cmnd	Operand (example)	Byte	Function	V[1]	C[1]	Z[1]
=	Assign value (bit or byte)					
	O00.00	4	1-bit address			
	BM00.00	4	8-bit address			
	O00.00[10]	8	1-bit address with constant offset		n	n
	O00.00[BM01.00]	14	1-bit address with variable offset			
	BM00.00[10]	8	8-bit address with constant offset			
	BM00.00[BM01.00]	14	8-bit address with variable offset			
=N	Assign negated value (bit or byte)					
	O00.00	16	1-bit address			
	BM00.00	16	8-bit address			
	O00.00[10]	20	1-bit address with constant offset		n	d
	O00.00[BM01.00]	26	1-bit address with variable offset			
	BM00.00[10]	12	8-bit address with constant offset			
	BM00.00[BM01.00]	18	8-bit address with variable offset			
=D	Assign value (double byte) [1]					
	BM00.00	4	Even address			
	BM00.01	18	Odd address		n	n
	BM00.00[10]	16	16-bit address with constant offset			
	BM00.00[BM01.00]	22	16-bit address with variable offset			
=DW	Assign value (double word)					
	BM00.00		Even address			
	BM00.01		Odd address	n	n	n
	BM00.00[10]		32-bit address with constant offset			
	BM00.00[BM01.00]		32-bit address with variable offset			

[1] Manipulation of **C**arry and **Z**ero flag:

- n no change of flag
- d defined change of flag
- u undefined change of flag

[1] See Swap commands (→ 4.3.5.7)

4.3.5.6 Set Commands

Cmnd	Operand (example)	Byte	Function	C[1]	Z[1]
	Set / reset with condition				
S	O00.00	12	1-bit address	n	d
R	O00.00	10	1-bit address		
	Set / reset without condition				
=1	O00.00	16	1-bit address	n	d
=0	O00.00	16	1-bit address		

[1] Manipulation of **C**arry and **Z**ero flag:

n no change of flag
d defined change of flag
u undefined change of flag

4.3.5.7 Swap Commands (Reverse Byte Sequence)

These commands let you reverse (swap) the sequence of bytes in word operations:

Cmnd	*Operand (example)*	*Byte*	*Function*	*C*[1]	*Z*[1]
	Load word swapping high byte and low byte				
LDS	BM00.00		Even address	?	?
	BM00.01		Odd address		
	Assign word swapping high byte and low byte				
=DS	BM00.00		Even address	?	?
	BM00.01		Odd address		

[1] Manipulation of **C**arry and **Z**ero flag:
- n no change of flag
- d defined change of flag
- u undefined change of flag

4.3.6 Arithmetic Commands

4.3.6.1 Addition and Subtraction

Cmnd	Operand (example)	Byte	Function	V[1]	C[1]	Z[1]
	Add 8-bit values					
ADD	BM00.00	4	8-bit address		d	d
	100	4	8-bit constant			
ADDD	Add 16-bit values					
	BM00.00	4	16-bit address (even address)		d	d
	BM00.01	10	16-bit address (odd address)			
	1000	4	16-bit constant			
ADDDW	Add 32-bit values					
	BM00.00		32-bit address (even address)	d	d	d
	BM00.01		32-bit address (odd address)			
	1000		32-bit constant			
SUB	Subtract 8-bit values					
	BM00.00	4	8-bit address		d	d
	100	4	8-bit constant			
SUBD	Subtract 16-bit values					
	BM00.00	4	16-bit address (even address)		d	d
	BM00.01	10	16-bit address (odd address)			
	1000	4	16-bit constant			
SUBDW	Subtract 32-bit values					
	BM00.00		32-bit address (even address)	d	d	d
	BM00.01		32-bit address (odd address)			
	1000		32-bit constant			

[1] Manipulation of O**V**erflow, **C**arry and **Z**ero flag: n no change of flag
d defined change of flag
u undefined change of flag

4.3.6.2 Multiplication and Division

Cmnd	Operand (example)	Byte	Function	V[1]	C[1]	Z[1]
	Multiply 8-bit values					
MUL	BM00.00		8-bit address		d	d
	100		8-bit constant			
MULD	Multiply 16-bit values					
	BM00.00		16-bit address (even address)		d	d
	BM00.01		16-bit address (odd address)			
	1000		16-bit constant			
MULDW	Multiply 32-bit values					
	BM00.00		32-bit address (even address)	d	d	d
	BM00.01		32-bit address (odd address)			
	1000		32-bit constant			
DIV	Divide 8-bit values					
	BM00.00		8-bit address		d	d
	100		8-bit constant			
DIVD	Divide 16-bit values					
	BM00.00		16-bit address (even address)		d	d
	BM00.01		16-bit address (odd address)			
	1000		16-bit constant			
DIVDW	Divide 32-bit values					
	BM00.00		32-bit address (even address)	d	d	d
	BM00.01		32-bit address (odd address)			
	1000		32-bit constant			

[1] Manipulation of O**V**erflow, **C**arry and **Z**ero flag: n no change of flag
d defined change of flag
u undefined change of flag

4.3.7 Comparison Commands

4.3.7.1 Comparison with Ensuing Jump

Cmnd	*Operand (example)*	*Byte*	*Function*	*V*[1]	*C*[1]	*Z*[1]
	Compare 8-bit values					
CMP	BM00.00		8-bit address		d	d
	100		8-bit constant			
CMPD	Compare 16-bit values					
	BM00.00		16-bit address (even address)		d	d
	BM00.01		16-bit address (odd address)			
	1000		16-bit constant			
CMPDW	Compare 32-bit values					
	BM00.00		32-bit address (even address)	d	d	d
	BM00.01		32-bit address (odd address)			
	1000		32-bit constant			

[1] Manipulation of O**V**erflow, **C**arry and **Z**ero flag: n no change of flag
d defined change of flag
u undefined change of flag

The following jump commands are possible responses to the result of the comparison:
JP=, JP<>, JP<, JP>, JP<=, JP>=

4.3.7.2 Comparison with Ensuing Logic Command

Cmnd	Operand (example)	Byte	Function	C[1]	Z[1]
	Compare for "equal" (evaluate as logical 1/0)				
CMP =	BM00.00	16	8-bit address	d	d
	100	16	8-bit constant		
CMP D=	BM00.00	18	16-bit address (even address)		
	BM00.01	22	16-bit address (odd address)		
	1000	16	16-bit constant		
	Compare for "not equal" (evaluate as logical 1/0)				
CMP <>	BM00.00	16	8-bit address	d	d
	100	16	8-bit constant		
CMP D<>	BM00.00	18	16-bit address (even address)		
	BM00.01	22	16-bit address (odd address)		
	1000	16	16-bit constant		
	Compare for "smaller than or equal to" (evaluate as logical 1/0)				
CMP <=	BM00.00	16	8-bit address	d	d
	100	16	8-bit constant		
CMP D<=	BM00.00	18	16-bit address (even address)		
	BM00.01	22	16-bit address (odd address)		
	1000	16	16-bit constant		
	Compare for "greater than or equal to" (evaluate as logical 1/0)				
CMP >=	BM00.00	16	8-bit address	d	d
	100	16	8-bit constant		
CMP D>=	BM00.00	18	16-bit address (even address)		
	BM00.01	22	16-bit address (odd address)		
	1000	16	16-bit constant		

[1] Manipulation of **C**arry and **Z**ero flag:
- n no change of flag
- d defined change of flag
- u undefined change of flag

The result leads to logic commands (A, AN, O, =, S...) or jump commands (JPC, JPCN).
➔ true: logical 1, false: logical 0

4.3.8 Shift and Rotation Commands

4.3.8.1 Shift Commands

Cmnd	Operand (example)	Byte	Function	C[1]	Z[1]
	Shift the bits in the processor accu				
LSL	Processor accu	2	Shift 8-bit accu to the left	d	d
LSR		6	Shift 8-bit accu to the right		
LSLD		2	Shift 16-bit accu to the left	d	d
LSRD		2	Shift 16-bit accu to the right		
	Shift bits in the operand				
LSLM	BM00.00	10	Shift 8-bit address to the left	d	d
LSRM		10	Shift 8-bit address to the right		
LSLDM	BM00.00	10	Shift even 16-bit address to the left		
	BM00.01	14	Shift odd 16-bit address to the left		
LSRDM	BM00.00	10	Shift even 16-bit address to the right		
	BM00.01	14	Shift odd 16-bit address to the right		

[1] Manipulation of **C**arry and **Z**ero flag:
- n no change of flag
- d defined change of flag
- u undefined change of flag

The Carry Bit will contain the value carried forward which will be lost after the next clock pulse.

4.3.8.2 Rotation Commands

Cmnd	Operand (example)	Byte	Function	C[1]	Z[1]
	Shift the bits in the processor accu				
ROL	<no operand>	2	Roll 8-bit accu to the left	d	d
ROR		10	Roll 8-bit accu to the right		
ROLD		2	Roll 16-bit accu to the left	d	d
RORD		20	Roll 16-bit accu to the right		
	Shift bits in the operand				
ROLM	BM00.00	10	Roll 8-bit address to the left	d	d
RORM		14	Roll 8-bit address to the right		
ROLDM	BM00.00	10	Roll even 16-bit address to the left		
	BM00.01	14	Roll odd 16-bit address to the left		
RORDM	BM00.00	26	Roll even 16-bit address to the right		
	BM00.01	34	Roll odd 16-bit address to the right		

[1] Manipulation of **C**arry and **Z**ero flag:

- n no change of flag
- d defined change of flag
- u undefined change of flag

The Carry Bit will contain the value carried forward which will be read after the next clock pulse.

4.3.9 Byte and Flag Manipulation

Cmnd	Operand (example)	Byte	Function	C[1]	Z[1]
	Increment (count up by 1)				
INC	BM00.00	4	8-bit address	n	d
INCD	BM00.00	10	Even 16-bit address		
	BM00.01	14	Odd 16-bit address		
	Decrement (count down by 1)				
DEC	BM00.00	4	8-bit address	n	d
DECD	BM00.00	10	Even 16-bit address		
	BM00.01	14	Odd 16-bit address		
	Set / clear carry bit				
SEC	<no operand>	2	Set carry bit = 1	d	n
CLC		2	Clear carry bit = 0		
	Other commands				
CLR	O00.00	4	Clear 1-bit address	n	n
	BM00.00	4	Clear 8-bit address		

[1] Manipulation of **C**arry and **Z**ero flag:

n no change of flag
d defined change of flag
u undefined change of flag

4.3.10 Module Calls

Program modules are called by special jump commands:

Cmnd	Operand (example)	Byte	Function	C[1]	Z[1]
	Call program module				
JPP	<Module name>	14	Jump	n	n
JPCP		18	Conditional jump (if logical 1)		
JPPI	<Byte marker>		Indexed jump to module <number>	?	?
	Call function module				
JPF	<Module name>	18	Jump	n	n
JPCF		26	Conditional jump (if logical 1)		
	Call KUBES module				
JPK	<Module name>	18	Jump	n	n
JPCK		26	Conditional jump (if logical 1)		
	Call initialisation module				
JPINIT	<Module name>	14	Jump	n	n

[1] Manipulation of **C**arry and **Z**ero flag:

- n no change of flag
- d defined change of flag
- u undefined change of flag

4.3.11 End-of-Module Commands

Immediately quit the module:

Cmnd	Operand (example)	Byte	Function	C[1]	Z[1]
RET	None		Return (quit module)	n	n
RETB			Conditional return (quit module if log. 1)		

[1] Manipulation of **C**arry and **Z**ero flag:

- n no change of flag
- d defined change of flag
- u undefined change of flag

4.3.12 Jump Commands

Cmnd	Operand (example)	Byte	Function	C[1]	Z[1]
	General jumps				
JP	<Jump label>	4	Unconditional jump	n	n
JPC		8	Conditional jump (if logical 1)		
JPCN		8	Conditional jump (if logical 0)		
	Jumps after comparison (→ 4.3.7.1)				
JP=	<Jump label>	4	Jump if equal	n	n
JP<>		4	Jump if not equal		
JP<		4	Jump if smaller		
JP>		4	Jump if greater		
JP<=		4	Jump if smaller than or equal to		
JP>=		4	Jump if greater than or equal to		
	Jumps depending on carry, zero or overflow flag				
JPCS	<Jump label>	4	Jump if Carry = 1	n	n
JPCC		4	Jump if Carry = 0 (no carry)		
JPZS		4	Jump if Zero bit = 1 (result = zero)		
JPZC		4	Jump if Zero bit = 0 (result not zero)		
JPV			Jump if Overflow bit = 1 (overflow = yes)		
	Jumps depending on the value (two's complement)				
JP+	<Jump label>	4	Jump if positive	n	n
JP-		4	Jump if negative		

[1] Manipulation of **C**arry and **Z**ero flag:
- n no change of flag
- d defined change of flag
- u undefined change of flag

4.3.13 Sequential Function Chart Commands

Help to structure programs as sequential function charts:

Cmnd	Operand (example)	Byte	Function	C[1]	Z[1]
	Jump labels				
JPSTEP	<Byte marker>		Go to step <number>	?	?
	Step				
STEP	<Number>		Start of step	?	?

[1] Manipulation of **C**arry and **Z**ero flag:

- n no change of flag
- d defined change of flag
- u undefined change of flag

4.3.14 Copy Commands

Cmnd	Operand (example)	Byte	Function	C[1]	Z[1]
	Copy bits				
C1T8	I00.00	16	1-bit addresses into 8-bit accu	n	n
C8T1	O00.00	16	8-bit accu into 1 bit addresses		
C1T16	M00.00	16	1-bit addresses into 16-bit accu		
C16T1	SM00.00	16	16-bit accu into 1 bit addresses		

[1] Manipulation of **C**arry and **Z**ero flag:
n no change of flag
d defined change of flag
u undefined change of flag

4.3.15 BCD Commands

Cmnd	Operand (example)	Byte	Function	C[1]	Z[1]
	Binary <-> BCD (conversion in processor accu)				
BINBCD3	<No operand>	4	Binary to BCD conversion(3 decades)	d	d
BCDBIN3		4	BCD to binary conversion(3 decades)		

[1] Manipulation of **C**arry and **Z**ero flag:

- n no change of flag
- d defined change of flag
- u undefined change of flag

4.3.16 Programmable Pulses

Cmnd	Operand (example)	Byte	Function	C[1]	Z[1]
	Edge detection				
=	PP00.00	16	Rising edge	d	d
=N		16	Falling edge		
	Logical operation with pulse signal				
L	PP00.00	4	Load	n	d
A		4	And		
O		4	Or		
LN		6	Load negated value		
AN		8	And negated value		
ON		8	Or negated valuet		

[1] Manipulation of **C**arry and **Z**ero flag:

- n no change of flag
- d defined change of flag
- u undefined change of flag

4.3.17 Programmable Timers

Cmnd	Operand (example)	Byte	Function	C[1]	Z[1]
	Start timer (condition: logical 1 = start, logical 0 = stop) [2]				
=	PT00.00:100*10ms:E:R	14	With constant time value	d	d
=	PT00.00:BM00.00*10ms:E:R	20	With variable time value		
=N	PT00.00:100*10ms:E:R	18	With constant time value		
=N	PT00.00:BM00.00*10ms:E:R	24	With variable time value		
	Halt timer (condition: logical 1 = halt, logical 0 = resume)				
=TH	PT00.00	14	Halt (current value is retained)	d	d
	Logical operation with timer output				
L	PT00.00	4	Load	n	d
A		4	And		
O		4	Or		
LN		6	Load negated signal		
AN		8	And negated signal		
ON		8	Or negated signal		

[1] Manipulation of **C**arry and **Z**ero flag:
- n no change of flag
- d defined change of flag
- u undefined change of flag

[2] „:R“: If you place this switch at the end, the times will be stored in the NV-RAM as remanent data when you start a timer (→ 3.5.3)

4.3.18 Programmable Counters

Cmnd	Operand (example)	Byte	Function	C[1]	Z[1]
	Set counter (condition: logical 1 = start, logical 0 = stop) [2]				
=	C00.00:100:F:R	14	With constant count	d	d
=	C00.00:BM00.00:F:R	20	With variable count		
=N	C00.00:100:F:R	18	With constant count		
=N	C00.00:BM00.00:F:R	24	With variable count		
	Counting pulse (logical 1 = count)				
=C	C00.00	14	Integrated edge detection	d	d
	Logical operation with counter output				
L	PC00.00	4	Load	n	d
A		4	And		
O		4	Or		
LN		6	Load negated signal		
AN		8	And negated signal		
ON		8	Or negated signal		

[1] Manipulation of **C**arry and **Z**ero flag:
- n no change of flag
- d defined change of flag
- u undefined change of flag

[2] ":R": If you place this switch at the end, the counters will be stored in the NV-RAM as remanent data when you set a counter (→ 3.5.3)

4.3.19 Special Commands

Cmnd	Operand (example)	Byte	Function	C[1]	Z[1]
	Dummy instruction (placeholder)				
NOP	<no operand>	2	Dummy instruction	n	n
	Safety functions				
O_OFF	-		Switch off power element of all outputs	u	u
O_ON	-		Switch on power element of all outputs	u	u
RESET	-		Reset all non-remanent markers, timers, counters and stop program	u	u
WAIT	n		Wait loop (only available for interrupt module 17, low voltage fault, → 7.5.3) loop time = n (1...6) * 10 ms	u	u

[1] Manipulation of **C**arry and **Z**ero flag:

- n no change of flag
- d defined change of flag
- u undefined change of flag

4.3.20 Initialisation Module Commands

Initialisation modules are a special category of modules. None of the commands described earlier in this chapter can be used in initialisation modules. On the other hand, the commands described below apply to the initialisation modules only.

Operand	Data Type	Value	Byte	Allocation of Values
Assign bit value				
O00.00	BIT	1		to an address
		1,0,1,1...		Var. values to several addresses
		[16],1		Same value to several addresses
Assign byte value (8 bit)				
BM00.00	BYTE	75		Decimal value to an address
		1,18,0,125...		Var. values to several addresses
		[8],128		Same value to several addresses
		"Kuhnke“		ASCII value to an address
Assign word value (16 bit)				
BM00.00	WORD	19285		Decimal value to an address
		1,18,0,125,63000		Var. values to several addresses
		[8],13283		Same decimal value to several addresses
Assign double word value (32 bit)				
BM00.00	DWORD	192853		Decimal value to an address
		1,18,0,125,163000		Var. values to several addresses
		[8],132834		Same decimal value to several addresses
Assign text string				
BM00.00	TEXT	"Kuhnke“		A string of text
		"ProfiControl“, "PLC“		Several strings of text

4.3.21 Data Module Commands

Cmnd	Operand (example)	Byte		Function	C[1]	Z[1]
	Load data module (data block) [2]					
LoadDB	x,<name>	12		Load data module <name> into DBx00.00...15.15 (x = prefix of operand)	d	d
	Byte1,<name>			Load data module <name> into DBx00.00...15.15 (x = value of byte 1)		
	x,byte2			Load data module number y into DBx00.00... 15.15 (y=value of byte 2, x=prefix of operand)		
	Byte1,byte2			Load data module number y into DBx00.00... 15.15 (y=value of byte 2, x=value of byte 1)		
	Store data module (data block) [2]					
StoreDB	x,<name>	12		Store DBx00.00...15.15 in data block <name> (x = prefix of operand)	d	d
	Byte1,<name>			Store DBx00.00...15.15 in data block <name> (x = value of byte 1)		
	x,byte2			Store DBx00.00...15.15 in data block number y (x=prefix of operand, y=value of byte 2)		
	Byte1,byte2			Store DBx00.00...15.15 in data block number y (x=value of byte 1, y=value of byte 2)		

[1] Manipulation of **C**arry and **Z**ero flag:
- n no change of flag
- d defined change of flag
- u undefined change of flag

[2] x = 0...7 Group number in data processing range DBx00.00...
y = 1...255 number of data block

4.4 Registers

The control unit knows how to handle four **types of operands** of different sizes:

- 1-bit operands
- 8-bit operands (bytes)
- 16-bit operands (words)
- 32-bit operands (double words)

The **accumulator** in the CPU can be used as 1-bit, 8-bit or 16-bit register.

- 1-bit operands are used for internal byte operations; only bit 7 of the 8-bit accu is looked at, though.
- 16-bit operands are mapped to a 16-bit accu which uses the above 8-bit accu as its low byte. The processing of word operands is set off by commands suffixed by a "D".
- 32-bit operations are sequences of 4 subsequent byte operands. Double word operations occur as load, assignment and arithmetic commands as well as the initialisation command in the init module.

To prevent faulty computations, we recommend not to mix different types of operands in operations that belong together.

4.5 Addressing

There are 2 ways of assigning the value of operands:

- Absolute value (constant, voltage or current value)
- Contents of an operand (bit, byte, word)

4.5.1 Address Mnemonic

The actual operand addresses are indicated as mnemonics such as BM00.00, O00.00, PT00.00. The processor's actual address management remains invisible.

The command line

```
LBM    BM00.00
```

means that the contents of a memory location is to be loaded whose mnemonic name is "BM00.00".

4.5.2 Offset Addressing

The absolute addresses of the local operands can be supplemented by an offset. In that case, the absolute address and the offset will be added to make up the actual address.

The command line

```
L      BM00.00[BM00.01]
```

means that the value in BM00.01 (offset) will be added to the address of BM00.00. The load instruction will then refer to the new address made up of the other two addresses.

Choose an offset that takes the applicable operand range (max. 256 addresses) into account.

Reason:

If the new operand exceeds the valid operand range, the program will read (L, A, O... instructions) or write (assignment commands =, =N, S...) an operand from another range. This may start unintended machine functions or destroy the program.

Examples:

```
L     I00.00[5]    is the same as
L     I00.05
```

```
=     O01.00[6]    is the same as
=     O01.06
```

```
=     BM01.00[17]  is the same as
=     BM02.01
```

4.5.3 Types of Addressing (Example: Load Command)

Load contents of an operand

```
L     I00.00                  1-bit address
L     BM00.00                 8-bit address
LD    BM00.00                 16-bit address
```

Load constant value

8-bit constant (0...255):

```
L     100                     decimal
L     $64                     hexadecimal
L     %01100100               binary
L     'A'                     ASCII
```

16-bit constant (0...65535):

```
LD    10000                   decimal
LD    $3FEA                   hexadecimal
LD    %0010011100010000       binary
LD    'AB'                    ASCII
LD    4.5V                    voltage (-10...+10V)
LD    5mA                     current (-20...+20mA)
```

Load contents of an operand addressed with an offset

with constant offset (0...255):

```
L     M00.00[10]              1-bit address
L     BM00.00[10]             8-bit address
LD    BM00.00[10]             16-bit address
```

with variable offset (0...255):

```
L     M00.00[BM00.01]         1-bit address
L     BM00.00[BM00.01]        8-bit address
LD    BM00.00[BM00.01]        16-bit address
```

4.6 Special Functions of Internal I/Os

The internal I/Os can be used for special functions which will be described in this section.

The special functions are provided by software modules that are part of the operating system.

4.6.1 Transfer Memory

A transfer memory in address range SLQ00.00...15.15 interfaces between the I/Os and the user program.

4.6.1.1 Selection of Functions

Operand		Bit								Comment
Address	Symbol	7	6	5	4	3	2	1	0	
Op. mode of first group of 4 inputs (I00.00...03)										
SLQ00.10	MODE_1	0	0	0	0	0	0	0	0	Standard inputs
		0	0	0	0	0	0	0	1	Interrupt inputs (→ 4.6.2)
		0	1	0	0	0	0	0	x	Input state will also be sent to bus 1
		1	0	0	0	0	0	0	x	Input state will also be sent to bus 2
Op. mode of second group of 4 inputs (I00.04...07)										
SLQ00.11	MODE_2	0	0	0	0	0	0	0	0	Standard inputs
		0	0	0	0	0	0	0	1	Interrupt inputs (→ 4.6.2)
		0	1	0	0	0	0	0	x	Input state will also be sent to bus 1
		1	0	0	0	0	0	0	x	Input state will also be sent to bus 2

Operand		Bit								Comment
Address	Symbol	7	6	5	4	3	2	1	0	
Op. mode of first group of 4 outputs (O00.00...03)										
SLQ00.12	MODE_3	0	0	0	0	0	0	0	0	User program will actuate the outputs
		0	0	0	0	0	0	0	1	A partner station on bus 1 will actuate the outputs
		0	0	0	0	0	0	1	0	A partner station on bus 2 will actuate the outputs
Op. mode of second group of 4 outputs (O00.04...07)										
SLQ00.13	MODE_4	0	0	0	0	0	0	0	0	User program will actuate the outputs
		0	0	0	0	0	0	0	1	A partner station on bus 1 will actuate the outputs
		0	0	0	0	0	0	1	0	A partner station on bus 2 will actuate the outputs
SLQ00.14										
SLQ00.15	VERSION									Software version

4.6.2 Interrupt Function of Internal Inputs

The function is set in transfer addresses SLQ00.10 (I00.00…03) and SLQ00.11 (I00.04…07).

4.6.2.1 Interrupt Modules

Every interrupt automatically starts an interrupt module containing the user's instructions as to how to react to the interrupt. The processor will continue to map the input states to operands I00.00…07.

The interrupt modules are allocated as follows:

Input	Interrupt Module No.
I00.00	24
I00.01	25
I00.02	26
I00.03	27
I00.04	28
I00.05	29
I00.06	30
I00.07	31

The next two sections detail the applicable transfer addresses and their assignment.

4.6.2.2 Transfer Addresses of Inputs I00.00...03

Address	Symbol	Function		
SLQ00.00	INT_LH_0	Interrupt caused by: rising edge (low -> high)	I00.00	CPU writes 255 if interrupt was caused by this channel. User program runs the defined interrupt module to respond.
SLQ00.01	INT_LH_1		I00.01	
SLQ00.02	INT_LH_2		I00.02	
SLQ00.03	INT_LH_3		I00.03	
SLQ00.04	INT_HL_0	Interrupt caused by: falling edge (high -> low)	I00.00	
SLQ00.05	INT_HL_1		I00.01	
SLQ00.06	INT_HL_2		I00.02	
SLQ00.07	INT_HL_3		I00.03	
SLQ01.00	ENI_LH_0	Interrupt trigger enabled: rising edge (low -> high)	I00.00	User program writes 255 to enable the source of the interrupt, or 0 to disable it. The setting is transferred to the CPU by 255 in SLQ01.14
SLQ01.01	ENI_LH_1		I00.01	
SLQ01.02	ENI_LH_2		I00.02	
SLQ01.03	ENI_LH_3		I00.03	
SLQ01.04	ENI_HL_0	Interrupt trigger enabled: falling edge (high -> low)	I00.00	
SLQ01.05	ENI_HL_1		I00.01	
SLQ01.06	ENI_HL_2		I00.02	
SLQ01.07	ENI_HL_3		I00.03	
SLQ01.15	SET_ENI	User program writes 255 to transfer new settings in SLQ01.00...07 to the CPU. CPU acknowledges by writing 0.		

4.6.2.3 Transfer Addresses of Inputs I00.04...07

Address	Symbol	Function		
SLQ02.00	INT_LH_4	Interrupt caused by: rising edge (low -> high)	I00.04	CPU writes 255 if interrupt was caused by this channel. User program runs the defined interrupt module to respond.
SLQ02.01	INT_LH_5		I00.05	
SLQ02.02	INT_LH_6		I00.06	
SLQ02.03	INT_LH_7		I00.07	
SLQ02.04	INT_HL_4	Interrupt caused by: falling edge (high -> low)	I00.04	
SLQ02.05	INT_HL_5		I00.05	
SLQ02.06	INT_HL_6		I00.06	
SLQ02.07	INT_HL_7		I00.07	
SLQ03.00	ENI_LH_4	Interrupt trigger enabled: rising edge (low -> high)	I00.04	User program writes 255 to enable the source of the interrupt, or 0 to disable it. The setting is transferred to the CPU by 255 in SLQ03.14
SLQ03.01	ENI_LH_5		I00.05	
SLQ03.02	ENI_LH_6		I00.06	
SLQ03.03	ENI_LH_7		I00.07	
SLQ03.04	ENI_HL_4	Interrupt trigger enabled: falling edge (high -> low)	I00.04	
SLQ03.05	ENI_HL_5		I00.05	
SLQ03.06	ENI_HL_6		I00.06	
SLQ03.07	ENI_HL_7		I00.07	
SLQ03.15	SET_ENI2	User program writes 255 to transfer new settings in SLQ03.00...07 to the CPU. CPU acknowledges by writing 0.		

5 CANopen

5.1 What is CANopen?

At this point, we will only briefly touch upon the key properties of CANopen.

For further information please refer to instruction manual E 615 GB, CANopen.

CAN is an acronym for "Controller Area Network". The "open" suffix indicates that this is a standardised protocol variant which allows devices of different manufacturers to exchange data.

- International standard (layers 1/2) to DIN ISO 11898
- Data exchange via earthed 2-wire cables
- Transfer rates up to 1 Mbit/s
- No real master/slave approach (→ 5.2.1)
- Data exchange between any two stations
- Data transfer methods: peer-to-peer, broadcast, or multicast
- Frame priority set by ID
- Cheap due to high utilisation rate in the automotive industry

5.2 CanControl 691 and CANopen

CanControl 691 features two bus ports: bus 1 and bus 2. Whereas bus 2 is a CANopen port in all variants, bus 1 can be either a CANopen or a PROFIBUS port, depending on the model variant (→ 6).

5.2.1 Master/Slave?

CANopen has no masters and slaves in the true sense of the word, at least not where the exchange of process data is concerned. However, there is to be one network station that provides the so-called management functions which are detailed in the CANopen manual (→ E 615 GB).

Terminology

We would still like to use the terms master and slave:

- Master
  CanControl 691 is able to provide the above management functions. In that case it would act as a master.
- Slave
  The unit can also act as a simple station of a CANopen network. In that case we will refer to it as a slave.

5.2.2 CANopen Operands

5.2.2.1 Process Map of External Operands

External operands are read via CANopen as input states or actuated as outputs. There are different sources and destinations:

- Remote input/output devices
- Other programmable logic controllers (PLCs)
- Positioning controllers (motion controllers), etc.

The process map of the external operands has 1024 byte of output data and 1024 byte of input data available to it:

- Bus 1 (CanControl 691-C)
  operand ranges DNzxx.yy (→ 4.3.3.1)
- Bus 2
  operand ranges CNzxx.yy (→ 4.3.3.3)

5.2.2.2 CANopen Messages

Status

- Bus 1 (CanControl 691-C)
  operand ranges DSAxx.yy (→ 4.3.3.1)
- Bus 2
  operand ranges CSAxx.yy (→ 4.3.3.3)

Emergency

- Bus 1 (CanControl 691-C)
  operand ranges DEAxx.yy (→ 4.3.3.1)
- Bus 2
  operand ranges CEAxx.yy (→ 4.3.3.3)

For a detailed description of these messages refer to the CANopen manual (E 615 GB).

5.3 EDS Files

EDS files are Electronic Data Sheets that describe the unit. These files are required for configuring the CANopen network. Every member of the 691 family of products featuring its own CANopen port also has its own EDS file, which resides on the **Software & Information** CD-ROM.

- **CanControl 691**
  EDS file: Ku_691.eds

I/O extensions to the 691 range

- **CanControl 691 I/O 16DI**
  Part no.: 691.001.01
  EDS file: Ku_691_16di.eds
- **CanControl 691 I/O 8DI/8DO**
  Part no.: 691.001.02
  EDS file: Ku_691_8di8do.eds
- **CanControl 691 I/O 8DI/8DIO**
  Part no.: 691.001.03
  EDS file: Ku_691_8dio8di.eds
- **CanControl 691 I/O DO24 D-Sub/smc**
  Part no.s: 691.001.04 and 691.001.05
  EDS file: Ku_691_24do.eds

Universal EDS File for 691 family

- EDS file: Ku_691_uni.eds

6 PROFIBUS-DP

Bus 1 of CanControl 691 can be configured as a PROFIBUS-DP port. In this case, the unit is called CanControl 691-DP and is to be added to the bus as a DP slave. Bus 2 remains a CANopen port like in all variants of the range (→ 5).

6.1 What is PROFIBUS?

At this point, we will only briefly touch upon the key properties of PROFIBUS-DP.

For further information please refer to instruction manual E 611 GB, PROFIBUS-DP.

PROFIBUS is a field bus. Its name is an acronym for "Process Field Bus". PROFIBUS has been developed to network multiple control units while maintaining a link to the field level, i.e. to the sensors and actuators, via remote I/Os.

In the other direction, it interfaces with the control level where one or several computers are installed to control the overall process.

6.1.1 Bus Protocol

CanControl 691-DP supports the PROFIBUS-DP protocol.

Open communication

The principle of open communication is to ensure that devices of various manufacturers can exchange data. The field bus has been standardised and the standard published as Euro norm EN 50170.

6.1.2 Topology

PROFIBUS is constructed as a line. Using repeaters, you can also create a tree structure. The number of stations on any one segment is limited to 32. Where that is not enough, you can establish a second segment which is connected to the first segment by a repeater (bi-directional line amplifier).

Repeaters also count as physical stations so that the number of "real" stations on the line is reduced to 31 (or to 30 if you are using 2 repeaters).

Using up to 3 repeaters you can increase the number of bus stations to 122.

6.1.3 Station Address

Every logical PROFIBUS station is assigned its own station address that other stations use to access this station. Addresses range from 0 to max. 126.

6.1.4 Network Configuration

VEBES, Kuhnke's PROFIBUS network configuration manager, is used to build a PROFIBUS network with a Kuhnke controller serving as master.

VEBES defines the bus parameters, the stations and their communication pathways (→ instruction manual E 611 GB, PROFIBUS-DP).

6.1.5 Communication

By default, PROFIBUS communication is handled via the process map stored by the master. The process map is a map of the external operands. It is updated automatically, but not synchronised with the user program cycle.

6.2 CanControl 691-DP and PROFIBUS-DP

Still at the discussion stage when this manual went to press.

7 Error Handling by the PLC

The control unit more or less monitors itself. Errors are reported to the controller whose reactions depend on the seriousness of the problem.

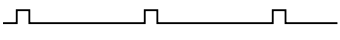
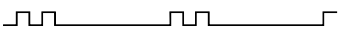
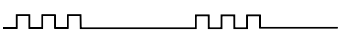
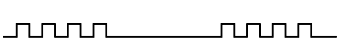
Errors and failures are numbered through (1 ... max. 255) and can be indicated in various ways.

7.1 Error and Failure Messages Overview

Error		Indication			
No.	*Description*	*"status" LED flashes n times*	*Error byte ERR00.00*		*Interrupt module [No.]*
1	Short-circuited output (→ 7.5.1).	1	1		18
-	System high voltage (→ 7.5.2)	No			
2	System low voltage (→ 7.5.3)	2	2		17
-	Low voltage dig. I/Os (→ 7.5.3)	No			21
-	Load 2 low voltage (→ 7.5.3)	No			22
3	Watchdog (→ 7.5.4)	3	No		
4	No communication (→ 7.5.5)	No	4		
5					
6					
8	Checksum error in user program (→ 7.5.6)	8	No		
9	Hierarchy error (→ 7.5.7)	9	No		
10					
11					
12	Short circuit (1) removed”	No	0		
13	Voltage back to normal	No	0		

7.2 Status LED

In case of a problem, the light-emitting diode labelled "status" will output a red flash code indicative of the error:

No.	Flash code
1	
2	
3	
4	
etc.	

The flashes pulse quickly (250/250 ms). Then there is a break of 1 s and the flash code is repeated.

7.3 Error Byte "ERR00.00"

The error code is written into error byte ERR00.00 which the user program can use for evaluation:

Example:

```
L      ERR00.00
C8T1   O00.00          ;binary indication
                       ; by 8 outputs
```

7.4 Interrupt Module [No.]

The error triggers an interrupt request (IRQ). The monitor program immediately runs the appropriate interrupt module for further handling.

7.5 Description of Errors

7.5.1 Short-circuited Output (Error No. 1)

Cause

- Short circuit
- Overload

Indication

- "status" LED flashes
- KUBES displays plain text message
- Error byte ERR00.00 will be set to 1

Reaction

- The faulty output will be switched off thermally.
- Interrupt module no. 18 will be started. In this module, the user can specify how the controller is to react:

Example (interrupt module 18):

```
O_OFF                          ;switch off all outputs
=1        Mxx.xx               ;set marker
```

The program will continue to run, but the outputs will be switched off on their outgoing side (their internal status will be retained although their LEDs will also go out).

7.5.1.1 Corrective action

Remove the short circuit, then

either

Reactivate the outputs via the program (do not go via the program of the interrupt module, because the module only runs once when the error occurs).
Example:

```
L       Mxx.xx          ;outputs off?
JPCN    RESUME          ; jump if not
L       Iyy.yy          ;input "short removed"
JPCN    RESUME          ; jump if not
O_ON                    ;reactivate outputs
=0      Mxx.xx          ;reset marker
RESUME  ....            ;normal program
```

- All outputs that are set internally will be reactivated, and the program will be resumed.
- "status" LED turns green
- Error byte "ERR00.00" is cleared

or

Restart the controller

- via the hardware: switch the supply off and back on again
- via the software: in KUBES, choose RESET, then RUN.

7.5.2 High Voltage

Supply voltage: 24 VDC ± 20%

An integrated voltage monitor reacts if set limits have been exceeded.

Cause

- Supply voltage up to approx. > 35 V

Reaction

- components will be destroyed

7.5.3 Low Voltage

7.5.3.1 System Power Supply (Error No. 2)

Supply voltage: 24 VDC ± 20%

An integrated voltage monitor reacts if the voltage drops below set limits. There is a two-stage response:

7.5.3.1.1 Stage 1

Cause

System power supply (→ 3.3) down to approx. < 19 V

Reaction

- Interrupt module no. 17 will be started.
- The program will not be interrupted as yet.

Buffered operands (markers, timers and counters) may be reset unintentionally if the user program continues to run at this stage. This could be caused by inputs detecting a 0 signal due to the voltage being low.

Indication

- "status" LED flashes
- Error byte ERR00.00 will be set to 2

7.5.3.1.2 Stage 2

Alternative 1

Cause

The voltage goes back up to 24 V DC ± 20%

Reaction

- "status" LED turns green
- Error byte "ERR00.00" is cleared
- The program runs on without interruption.

Alternative 2

Cause

Supply voltage goes down further to approx. < 17.5 V

Reaction

- The 5 V system voltage will be interrupted
- STOP: The program will be stopped
- RESET: Outputs, error byte ERR00.00, and unbuffered markers, timers and counters will be reset
- All LEDs will be off

7.5.3.1.3 Corrective action

As a precautionary measure in the user program to protect buffered operands:
The user program should be halted until the cause has reached its 2nd stage or until a practical wait loop has been completed.

Example (interrupt module no. 17):

```
WAIT    5               ;wait for 5 * 10 ms = 50 ms
L       ERR00.00        ;check error byte
CMP     2               ;voltage still low?
JP<>    RESUME          ; jump if not
RESET                   ; else RESET and Stop program
RESUME  NOP             ;resume program run
```

*The WAIT instruction produces a program loop whose length is defined as n * 10 ms. If this time is longer than approx. 70 ms, the system will detect a watchdog error and switch off the controller automatically. Practically feasible wait loops should be no longer than about 50 ms.*

7.5.4 Watchdog (Error No. 3)

Cause

- Runtime of a module is > 75 ms

or

- The runtime of the overall program is > 2 s

Indication

- "status" LED flashes
- KUBES displays plain text message

Reaction

- STOP: The program will be stopped
- RESET: Outputs, error byte ERR00.00, and unbuffered markers, timers and counters will be reset

Corrective action

Modify the program structure to reduce the program cycle.

Restart the controller:

- via the hardware: switch the supply off and back on again
- via the software: in KUBES, choose RESET, then RUN.

7.5.5 No Communication (Error No. 4)

This error message only applies to controllers that are part of a PROFIBUS network.

Cause

No communication with the partner stations, caused by:

- PROFIBUS cable failure
- Partner station failure
- Wrong station address
- Illegal baud rate

Reaction

- The values of the external inputs are no longer up to date.
- Indication
- Error byte ERR00.00 will be set to 4 or 5.

Corrective action

- Find and remove cause of problem

7.5.6 Checksum Error in User Program (Error No.8)

As the program is being compiled, the system applies a specific algorithm to the entire user program memory to generate a checksum (CS).

Cause

When you switch the controller on, the monitor program will generate another checksum and compare it with the one it saved. It will detect an error if the two sums do not match.

Reaction

- The controller will not start.

Indication

- "status" LED flashes
- Error byte ERR00.00 will be set to 8

Corrective action

- Find and remove cause of problem
- Transmit the project to the controller again (KUBES).

7.5.7 Hierarchy Error (Error No. 9)

The starting of the program or any of the modules must not exceed certain hierarchy limits (→ Programming manual E 417 GB, Module Programming).

At the programming stage, the controller will indicate a hierarchy error as soon as you transmit a program. At that point, this is just an alert that there might be an error. Only when the program starts will the controller check it for any hierarchy errors. The message will disappear if it does not find any.

Cause

When you switch the controller on or when KUBES sends a start command, the monitor program checks the user program for hierarchy errors (a module starts the module it was itself started by, or the module nesting depth exceeds 5 layers).

Reaction

- The controller will not start.
- Indication
- "status" LED flashes
- Error byte ERR00.00 will be set to 9

Corrective action

- Find and remove cause of problem
- Transmit the project to the controller again (KUBES).

8 Appendix

8.1 Technical Data

8.1.1 Basic Data

Type ..... compact plastic housing
- Protection ..... IP 65
- Dimensions L x W x H [mm] ..... 160 x 110 x 24
- Weight ..... 520 g

Admissible ambient conditions
- Storage temperature ..... -25...+70 °C
- Ambient temp. during operation ..... 0...50 °C

Power supply
- System and inputs ..... 24 VDC ± 20%
- Outputs ..... 24 VDC ± 20%
- Connector ..... male round plug (in compliance with draft directive DR-303-1 of CANopen version 1.0 of 10 Oct. 1999)

Local status indication ..... LEDs [1]
- Status LED ..... red, green LED / Status
- Bus LED ..... yellow LED / bus communication

[1] Built-in LEDs: class 1 light emitting diodes (EN 60825-1)

Bus ports

Bus 1	CANopen
Baud rates	50...1000 kbit/s (software setting)
Potential separation	yes
Connectors	2 round plug connectors, W or ReverseKey coded (in compliance with draft directive DR-303-1 of CANopen version 1.0 of 10 Oct. 1999)
Bus in	male
Bus out	female
Station address	1...127 (software setting)
Bus 2	CANopen
Baud rates	50...1000 kbit/s (software setting)
Potential separation	yes
Connectors	1 round female plug connector with standard coding
Station address	1 ... 127 (software setting)

Inputs	8
Potential separation	no
Connector	round snap-on connector (8 mm), 3-pin, female
Indicators	green LEDs
Signal states	1: LED on, 0: LED off
Power consumption/input	max. 10 mA
Input delay	0.5 ms

Outputs	8
Potential separation	no
Connector	round snap-on connector (8 mm), 3-pin, female
Indicators	red LEDs
Switching states	1: LED on, 0: LED off
Current	0.5 A
Short-circuit protection	yes

8.2 Order Specifications

8.2.1 Controllers

CanControl 691-C
with 2 CANopen ports 691.000.00
CanControl 691-RS485
with 2 CANopen ports and
one RS485 communication port 691.000.99
CanControl 691-DP
with 1 CANopen port & 1 PROFIBUS port 691.000.01

8.2.2 I/O Modules

CanControl 691 I/O DI16 ..... 691.001.01
CanControl 691 I/O DI8/DO8 ..... 691.001.02
CanControl 681 I/O DI8/DIO8 ..... 691.001.03
CanControl 691 I/O DO24 (Kuhnke's MPP valve clusters) .. 691.001.04
CanControl 691 I/O DO24 smc (SMC valve clusters) 691.001.05

8.2.3 Accessories

CanControl 691, 8 mm cover caps 691.180.02

16 x 

CanControl 691, sealing set 691.180.01

16x 

2x 

2x 

1x 

2x 

2x 

17x 

Programming cable 691.180.01

8.2.3.1 Plug-type Power Supply Connectors

Make: Escha (www.escha.de)

Connector without cable

- Socket, axial ..... eurocon WASC4
- Socket, angular ..... eurocon WWASC4

Fully assembled connector, various cable lengths

- Socket, axial ..... eurocon WAK4
- Socket, angular ..... eurocon WWAK4

8.2.3.2 Plug-type I/O Connectors

Plug-type I/O connectors
Make: Escha (www.escha.de)
Fully assembled plug, various cable lengths
- Pins, axial ..... picocon SP3

8.2.3.3 Covers for Unused I/O Connectors

Make: Escha (www.escha.de)
- Cover MFK3

8.2.3.4 Plug-type Bus Connectors

CANopen bus connector
Make: Turck (www.turck.com)
Fully assembled plug, various cable lengths, female and male connector
- angular ..... 572 thin cable WSC-WKC
- straight (axial) ..... 572 thin cable RSC-RKC

8.3 References

8.3.1 Kuhnke Manuals

Title / Subject	*Number*
Programming manual Programming of Kuhnke's PLC Systems	E 417 GB
KUBES Programming software for Kuhnke's PLC Systems	E 327 GB
CANopen Basics and Configuration	E 615 GB
KUBES Modules	E 386 GB

8.3.2 CANopen Specifications (English Only)

Source

CAN in Automation
Am Weichselgarten 26
D-91058 Erlangen
Fax: 0049-9131-69086-79
E-mail: headquarters@can-cia.org
Web: http://www.can-cia.org

Title / Subject	*Number*
Application Layer and Communication Profile	DS 301 V4.01
Framework for Programmable CANopen Devices	DSP 302 V3.0
Device Profile for Generic I/O Modules	DS 401 V2.0
Device Profile Drives and Motion Control	DSP 402 V1.1
Device Profile Human Machine Interfaces	DSP 403 V1.0
Device Profile for IEC 1131 Programmable Devices	DSP 405 V1.0
Device Profile for Encoders	DSP 406 V2.0

8.4 Sales & Service

Please visit our Internet site to find a comprehensive overview of our sales and service network including all the relevant addresses. You are, of course, always welcome to contact our staff at the main factory in Malente or at sales headquarters in Neuhausen:

8.4.1 Main Factory in Malente

Kuhnke GmbH
Lütjenburger Str.101
D-23714 Malente
Phone +49-45 23-4 02-0
Fax +49-45 23-4 02-2 47
E-mail sales@kuhnke.de
Internet www.kuhnke.de

8.4.2 Sales Germany

Kuhnke GmbH
Sales Germany
Strohgäustr.3
D-73765 Neuhausen
Phone +49-71 58-90 74-0
Fax +49-71 58-90 74-80
E-mail sales@kuhnke.de
Internet www.kuhnke.de

8.5 Index