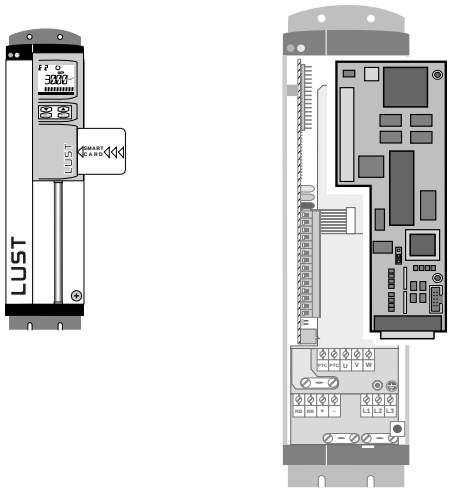
POSMOD1

EN

Positioning and Sequence Control for the MC6000 Servocontroller



Manual

Manual

for the POSMOD1 positioning and sequence control system

for servocontrollers of the MC6000 product line

Software versions:

- MC6000 Servocontroller: V150.30 [1)]
- POSMOD1: V1.03
- LUPOS User Interface: V1.00

[1)] The special V150.40 software supports the POSMOD1 positioning and sequence control system. The range of functions of the V150.40 version has been reduced by several elements in comparison with the V2.x standard software (see chapter 1.3).

Status: April 1997
Id. No.: 0792.21B.0 - 00

Dear customer!

Thank you for putting your trust in the Lust company with your purchase of a MASTERDRIVE drive system.

Installation and commissioning should be carried out by trained specialists. Please take time to carefully read this description and the Operating Instructions. If you observe all the instructions, you will save a lot of time and avoid problems.

It is also necessary to read the instructions because improper handling can damage the servodrive as well as other parts. The rotating parts of the drive and the high operating voltage also pose a risk of injury to persons.

If any other questions arise, you can reach us at the following address:

Lust Antriebstechnik GmbH
Gewerbestr. 5-9
D-35633 Lahnau
Germany
Telephone: +49 6441 966 -0
Fax: +49 6441 966 -137

The following pictograms are used in this description:

⇒ Caution! Danger of fatal injury from electrocution and rotating parts of the drive.

⇒ Attention! Observe instructions in all cases.

⇒ Attention! Before any repair or maintenance work, detach unit from power supply and wait approx. 2 minutes for the DC link capacitors to discharge.

⇒ Forbidden! Improper handling can lead to damage to the machine.

⇒ Useful note, tip.

Table of Contents

		Page
1	**General**	**1-1**
1.1	**Position plan**	**1-3**
1.2	**Commissioning**	**1-3**
1.3	**Changes from the standard software**	**1-6**
2	**Survey of the functions of the PosMod1**	**2-1**
2.1	**Operating modes**	**2-1**
2.2	**LuPos user interface:**	**2-2**
2.2.1	Programming and instruction set	2-3
2.2.2	Machine parameters	2-3
2.2.3	Data management commands	2-3
2.3	**Time behaviour of the position control system**	**2-4**
3	**Electrical connections**	**3-1**
3.1	**Overview of the connections**	**3-1**
3.2	**Designation key**	**3-2**
3.3	**PosMod1 inputs**	**3-3**
3.3.1	Electrical characteristics of the PosMod1 inputs	3-3
3.3.2	Functions for inputs	3-4
3.4	**PosMod1 outputs**	**3-6**
3.4.1	Electrical characteristics of the PosMod1 outputs	3-6
3.4.2	Functions for outputs	3-7
3.5	**Input and output wiring of the PosMod1**	**3-8**
3.6	**EKL300 terminal**	**3-9**
3.7	**Servocontroller inputs and outputs**	**3-11**
3.8	**I/O expansion with CKM100**	**3-13**
3.9	**CKM100 operations**	**3-15**
3.9.1	Status display	3-15
3.9.2	Resetting the CKM100 node module	3-15
3.9.3	Parameter display and settings	3-15
3.9.4	Node address of the CKM100	3-16
3.9.5	Baud rate of the CKM100	3-17
3.9.6	CKM100 error display	3-18
3.10	**Connection assignment (X16)**	**3-19**
4	**LuPos user interface**	**4-1**
4.1	**Installing and calling up LuPos**	**4-1**
4.2	**Layout and operations of LuPos**	**4-2**
4.3	**Configuration of LuPos**	**4-2**
4.4	**Menu structure**	**4-3**
4.4.1	Information menu	4-3
4.4.2	Programming menu	4-3
4.4.3	Setup menu	4-4
4.4.4	Diagnosis menu	4-6
4.4.5	Menu exit	4-7
4.5	**Program editor**	**4-7**
4.6	**Function key assignment**	**4-8**

Page

5 Machine parameters 5-1
5.1 Configuration of the inputs and outputs 5-1
5.2 Positioning parameters 5-5
5.3 CAN configuration 5-18
5.4 Definition of the reference run 5-19

6 Variables, markers and table positions 6-1
6.1 Variables 6-1
6.2 Markers 6-1
6.3 Table positions 6-2

7 Instruction set 7-1
7.1 Jump commands and subprogram call-ups 7-3
7.2 Setting command 7-7
7.3 Positioning commands 7-12
7.4 Wait commands and storing 7-15

8 Programming 8-1
8.1 Program layout 8-1
8.2 Program examples 8-2

9 Program and data management commands 9-1
9.1 Transferring and calling up programs and program blocks 9-1
9.2 Deleting programs and program blocks 9-2
9.3 Calling up the table of contents 9-2
9.4 Storing data in flash EPROM 9-3
9.5 Transferring and calling up machine parameters 9-3
9.5.1 Calling up parameters for sin² acceleration 9-3
9.5.2 Coded representation of inputs and outputs 9-4
9.6 Transferring and calling up variables 9-5
9.7 Transferring and calling up table positions 9-6
9.8 Transferring and calling up markers 9-7
9.9 Calling up status 9-8

10 Operations with LUSTBUS or CAN-Bus 10-1
10.1 Operations with LUSTBUS (RS485) 10-1
10.1.1 Select telegram 10-1
10.1.2 Telegram inquiry 10-1
10.1.3 Layout of use data 10-1
10.1.4 Example telegrams 10-2
10.2 Operations with CAN-Bus 10-4
10.2.1 Servocontroller parameter channel 10-4
10.2.2 Examples 10-5

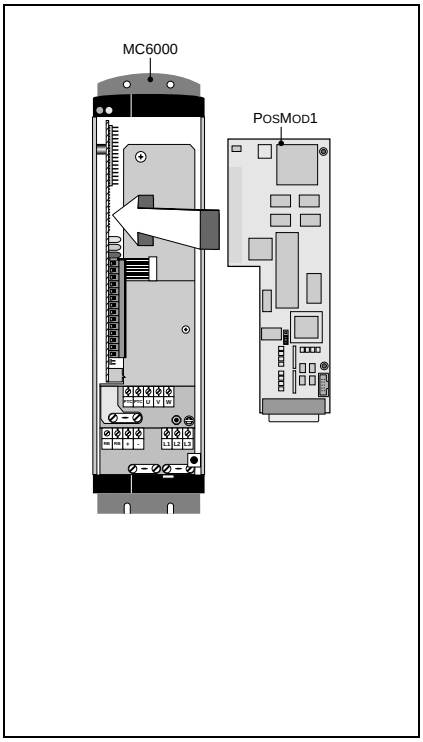
Appendix A Instruction set overview

Appendix B Error messages

Appendix C Technical data

Appendix D Table of key words

1 General



This description applies to the PosMod1 positioning and sequence control system which is available in some models for the MasterControl MC6000 product line of the MasterDrive MD6000 drive system. The control system is connected at optional slot 2 (see MC6000 operating instructions) and integrated into the servocontroller housing in this way.

The integration of the positioning control into the servocontroller has important advantages over the classical two-unit technique which consists of a servocontroller with torque control and a separate superimposed position control system with the position controlling functions.

Single unit technique:

Integration into the servocontroller facilitates commissioning and reduces wiring time. This lowers installation costs. Positioning drives using the single unit technique also have the advantage of higher reliability and greater resistance to interference due to their operating principle. The consistent agreement between the positioning module and the servocontroller, and the direct access to the system quuantitys current, rotational speed and position yields remarkable improvements in the control engineering characteristics of the drive system. The positioning module delivers a new set point every 5 ms which is then finely interpolated by the servocontroller so that the position controller receives a new set point every 500 µs.

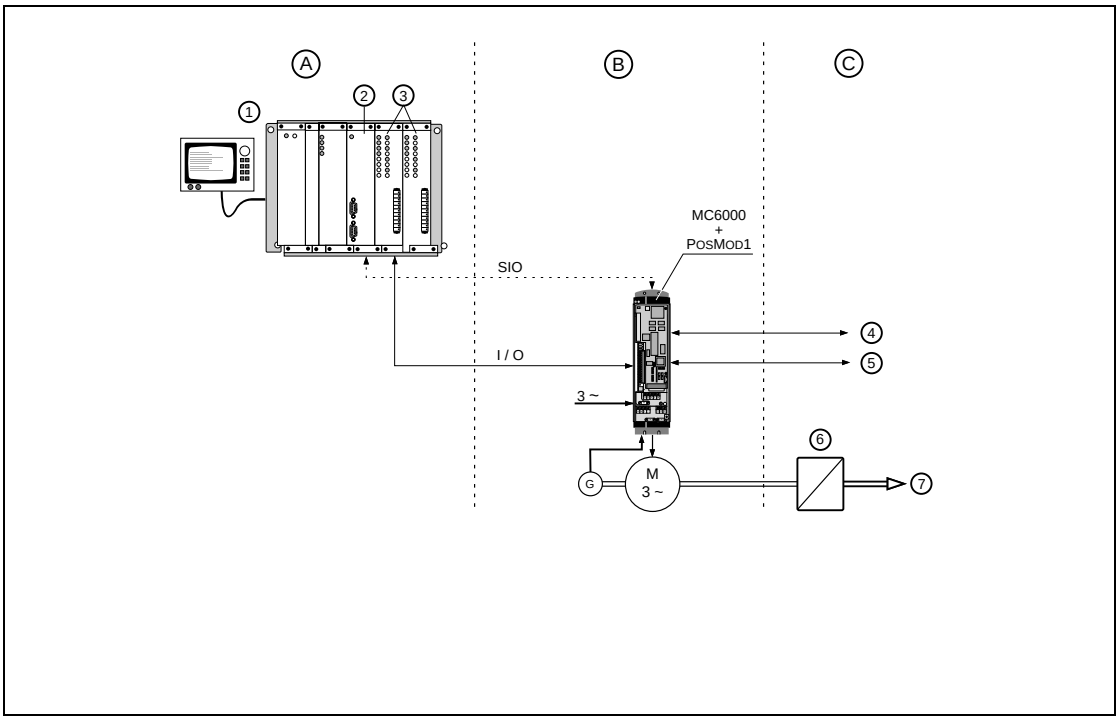
PosMod1 can also be used as a fully programmable sequence control system to execute time and position dependant rotational speed profiles.

The positioning module is designed for single axis applications as well as for multiple axis applications with limited requirements. The LuPos PC user interface in the SAA standard allows easy setting of parameters and programming of the controller. It also constantly visualises the current status of the controller and offers diagnostic functions. Up to 30 axes can be meshed with the RS-485 interface of the servocontroller, and joint parameters can be set with LuPos. The axes are coordinated with a superimposed controller (e.g. PLC, PC).

The option module is equipped with 8 inputs and 4 outputs which are PLC compatible and freely usable. The inputs and outputs of the MC6000 basic unit can be used as well.

The PosMod1 carries a 25-pin sub D connector on which the digital inputs and outputs are available. Wiring can be carried out either directly on this plug connector or on the EKL300 external terminal. With the CAN-Bus interface, the number of inputs and outputs can be increased with an extension module to up to 22 inputs and 14 outputs.

Design of a positioning drive with single unit technique:

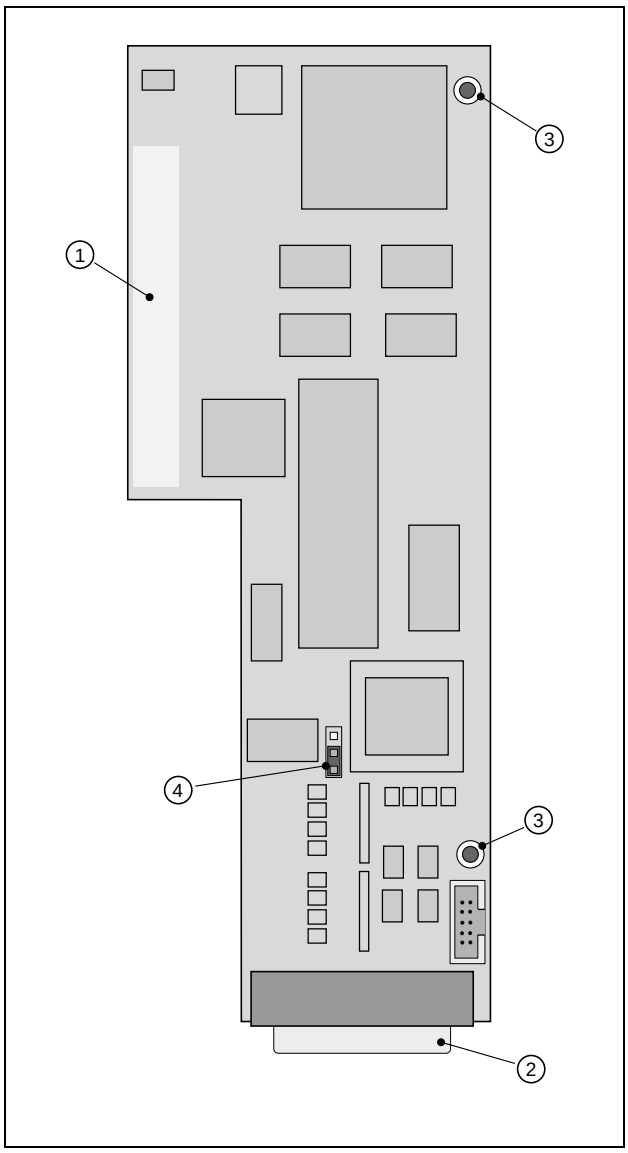


No.:	Function
A	Control level
B	Drive level
C	Process level
1	PLC with service unit
2	Serial interface SIO (RS485, InterBus-S, CAN-Bus)
3	I/O modules
4	Reference cam, safety limit switch
5	I/O signals out/ to process
6	Gears
7	Single process

An overview of the characteristics of PosMod1

- convenient LuPos PC user interface
- simple goal-oriented programming in program blocks
- 100 positioning programs with 3000 program blocks
- versatile instruction set
- 8 inputs (PLC-compatible, isolated, fully programmable)
- 4 outputs (capacity 500mA, PLC compatible isolated, fully programmable)
- Expansion module with 12 inputs and 8 outputs
- 9 different types of reference runs
- positioning with 4096 engine cycles without reference run due to Multiturn sensors (G3)
- 16 bit standard resolution (65536 increments)
- speed normalising with override (0-150%)
- absolute and relative positioning, unlimited motion (e.g. conveyor belts)
- linear or sinusoidal acceleration curve ($\sin^2$ shaped acceleration)

1.1 Position plan



No.	Function
1	Coupling on servocontroller (option slot 2)
2	Sub-D connector for wiring of the I/O (also EKL300 connection)
3	Bores for mechanical attachment
4	Jumper for reference cam input

Jumper	Reference cam on input
	IP17 (PosMod1) (factory setting)
	IS00 (MC6000-basic unit)

1.2 Commissioning

For commissioning a servocontroller with a PosMod1 optional module, the corresponding notes in the MC6000 Servocontroller Operating Instructions, chapter 3, always apply as well.

Control may not be activated until the servocontroller has been adapted to the engine by reading in the engine-specific SmartCard.

Without adaptation, the engine can come to a full stop and groan or rotate uncontrollably. This can damage the unit. A danger to persons is also possible!

Commissioning Procedure

What?	Why?
1. Read the MC6000 INSTUCTION MANUALand this description	Familiarise yourself with the machine. Avoid mistakes and danger from improper handling
2. ENPO input = low level (servocontroller control terminal strip)	Block power stage so that control does not become active immediately
3. Switch on external power supply	
4. Read in SMARTCARD (DRIVE area)	Adapt servocontroller to engine
5. Set parameter POENA = OFF (subject area _OPTN2) [1)]	Switch off POSMOD1 to set standard servocontroller parameters
6. Set SCJ (_SCON) moment of inertia parameter for unit	Adapt servocontroller to application in question
7. Set ENPO	Enable power stage
8. **TEST - only allowed with unlimited positioning distance** (e.g. engine uncoupled) Test drive system with the CTRL menu while torque-controlled (for use of CTRL menu, see: Operating Instructions, chapter 6, "The CTRL Menu".)	Is the drive unit running to your satisfaction? If the operation of the drive unit is not satisfactory, adapt and test the SCGFA (_SCON) parameter.
9. Further setting of the servocontroller is not necessary at this point [2)]	Reference input settings are done automatically [3)]
10. Set parameter POENA = ON (subject area _OPTN2, user level MODE = 4) [1)]	Switch POSMOD1 back on. POSMOD1 is recognised automatically.
11. **TEST with LUPOS USER INTERFACE:** Move drive unit with setup menu - manual; e.g. with +/- jog inputs	Test drive unit in position controlled mode.
12. Set POSMOD1 machine parameters	Axes setup with LUPOS user interface
• Set rotational speed limitation parameters K14 (and K12)	Rotational speed limited in application. Nominal rotational speed of engine not exceeded.
• Connect reference cam to IP17 input (see page 3-6). Select reference run type K70.	Specify reference run
13. **TEST with LUPOS user interface:** Test reference run in setup menu - reference run	Test reference run
• Set up further machine parameters	Configure inputs and outputs. Specify resolution for distance and acceleration.
14. Create user program	
15. Function test	

[1)] POENA can only be switched when POSMOD1 is neither in automatic mode (manual/automatic input) nor in manual mode (with the LUPOS user interface). The axis must not be located in the area of the limit switch.

Caution! With the settings POENA = OFF or = STBY it is not allowed to:

- use the LUPOS user interface (or the commands from chapter 9)
- Change the parameters in the subject areas _CONF or _REF of the servocontroller **with the serial interface**.

Otherwise, it is possible that POSMOD1 can no longer be activated! (**Solution:** Set unit back to factory settings).

[2] Possible application-specific adjustments to the configuration or control parameters can also be carried out after the function test.

[3] The servocontroller is equipped with its own set point channel (see illustration "Structure of Set Point Inputs", MC6000 Servocontroller Operating Instructions, chapter 7) The set point selectors are switched off by the automatic recognition unit of PosMod1 (RSSLx = RCON). If this is changed manually (RSSLx ≠ RCON), the E-PAR error message appears.

When the power stage is enabled (ENPO= high level), the control system is activated shortly after the mains voltage is switched on.

Setting servocontroller parameters - switch off control system

To set parameters for the MC6000 servocontroller (e.g. during commissioning), the position control must be switched off.

The effects of this action must be made clear: Switching off the position control switches off the current to the motor. This is not allowed in all applications without further measures (e.g. in lifting applications).

on servocontroller:	switch off control system:	input ENPO=0
	switch back on with:	input ENPO=1
with user interface:	switch off control system:	STOP 0 (setup menu, manual mode)
	switch back on with:	STOP B or STOP M.

As a user, you must make sure that no danger to persons or to the unit arises because the control system is switched off!

Note:
To determine if the control system is activated or not, one output of the servocontroller (OS00 or OS01) can be assigned a function "Active".

1.3 Changes from the standard software

The MC6000 servocontroller is equipped for PosMod1 operations with the V150.x modified software, which differs from the standard software in the following respects:

a) No support of I/O-Module 1 (AH1) and 1 channel analogue output (AH4).

b) No support of field weakening for asynchronous machines.

c) The following parameters are not available in the modified software, as they are not needed:
subject area _PCON: PCAMX
subject area _SCHON: field weakening parameters

d) The following functions of the servocontroller are intended for use as a speed control and may not be used with PosMod1.
- setup mode for setting the speed control
- direct input for external position control (speed function)
- Motorpoti function

2 Survey of the functions of the POSMOD1

2.1 Operating modes

The positioning module can be operated in various operating modes. The selection of operating modes is made either through positioning module inputs or through the LUPOS user interface.

1. Manual mode:
The designation "manual mode" is made in contrast to automatic mode. Within manual mode, one can distinguish between two sub-modes: setup mode and jog mode. Setup mode is selected with LUPOS; jog mode with inputs.

a) **Setup mode:**
In this mode, positioning and control commands, programs and other data are transferred over the serial interface of the MC6000. The axis can be given commands directly over the serial interface. Setup mode is used to determine and adapt machine parameters among other things. This operating mode is activated and deactivated by opening the "setup" menu.

a) **Jog mode:**
In jog mode, the axis can be moved manually in both directions by POSMOD1 inputs at the creep feed or rapid feed defined in the machine parameters. To do this, the Jog+ and Jog- functions must be assigned inputs, and keys must be attached to the inputs.

2. Automatic mode:
In automatic mode, a program located in the POSMOD1 positioning unit is executed. The desired program is selected either through the machine parameters or through inputs and started with the "Start" input. Automatic mode is selected with the input "Automatic".

3. Reference running:
Reference running can be seen as a special operating mode, because a reference run can be made in manual as well as in automatic mode. Encoders give information on the position within an engine revolution (as a rule). The reference run creates an absolute position relation (in relation to the complete axis) and must be done once after switching on.

A reference run is not necessary as a rule for infinite axes (e.g. conveyor belts) or for encoders which can determine the absolute position over several revolutions ("Multiturn", type G3).

Various reference run types and speeds are available which can be defined in the machine parameters. This is described in detail in chapter 5, Machine Parameters K70.

Note:
A reference run may not be interrupted!

2.2 LUPOS user interface

The LUPOS PC user interface primarily facilitates the creation of positioning programs by the user. An extensive, easy-to-learn instruction set is available. It is described completely in chapter 7, "Instruction set". Section 8, "Programming", contains an introduction using examples to program design and the use of the instruction set.

LUPOS also encompasses the functions for setting parameters, commissioning and diagnosis. Every axis is set up over the so-called machine parameters (parameter setting) This is done conveniently with a menu in the user interface. The commissioning of the axis and the positioning program is supported by LUPOS with the menu "Setup" and the menu "Diagnosis".

The functions of the user interface in key words:

- Creation and management of application programs, machine parameters, variables and tables; menu-controlled printing of the data
- Setup (parameter setting) of the axis (axes)
- Commissioning of the axis (axes)
- Monitoring and diagnosis
- Transmitting and receiving data to or from the servocontroller over the RS485 serial interface.

LUPOS uses pull-down menus for the various functions and supports the SAA standard for user interfaces. LUPOS can be run on IBM compatible PCs with at least 512 KB working memory under MS-DOS version 3.2 and above.

Notes on setting parameters:
Parameters for the servocontroller can be set with the help of KEYPAD, while the position control is operated with the LUPOS user interface.

		Setting parameters and storing:
Standard parameters (Servocontroller, engine, ...)	⇨	with KEYPAD, SMARTCARD and serial interfaces (RS485, InterBus-S, an-Bus)
Positioning parameters:	⇨	with serial interfaces, e.g. LUPOS (RS485)

The KEYPAD has no positioning function!

2.2.1 Programming and instruction set

The programming of the positioning module is carried out in a goal-oriented manner with simple commands.

The commands can be divided into various categories:

- Positioning commands
  - Moves to an absolute or relative position at a particular speed; indication of position and speed as a fixed value, a variable, or with a table.
  - Braking with maximum or programmed deceleration.
  - Reference running
- Jump commands
  - unconditional jump commands
  - conditional jump commands, dependant on the set or actual position, the status of an input, marker, counter, timer or a variable
- Subprogram call-ups
- Setting and loading commands
- Wait commands

2.2.2 Machine parameters

Machine and system parameters related to physical conditions and application data. They are entered once and, as a rule, are never changed again. The machine parameters are used to configure POSMOD1 and the driven axis (axes). Machine parameters are designated with "Kxx", where "xx" stands for a number.

The following settings, for example, are made with machine parameters:

- assigning functions to inputs and outputs
- Programming unit for distance, speed and acceleration
- max. speeds (moving, rapid feed and creep feed)
- max. acceleration and type of acceleration (linear or sin² shaped)
- Definition of the reference run
- Software limit switch

2.2.3 Data management commands

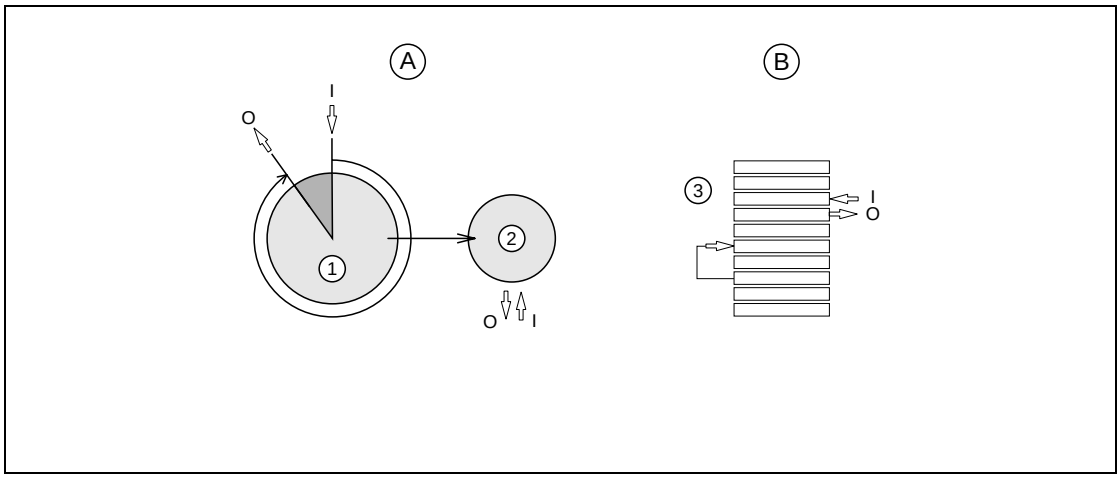
The commands for program and data management allow the loading and storing of positioning programs, machine parameters, etc. These commands can be carried out from the user interface in the menu "Programming". As an alternative, they can also be transmitted manually in setup mode.

The positioning unit memory can save up to 100 programs (P00 ... P99). They are saved in non-volatile memory (Flash EPROM).

2.3 Time behaviour of the position control system

PLC control systems work on a fixed cycle: First, the inputs are scanned, then the user program is executed, and finally the outputs are set appropriately.

As with other position control systems, PosMod functions according to a different principle. Programming is done in program blocks. The user program determines when inputs are scanned and outputs are set.



No.	Function
A	PLC with position control
B	PosMod1 positioning module
1	PLC program cycle
2	Positioning core
3	Programming in program blocks

This method has the advantage for programming that not every condition must be designated with a marker. The course of the program is controlled by jump commands and subprogram call-ups. This simplifies programming or allows for more complex programs. This method of operation corresponds to that of programming languages, e.g. BASIC.

The differences in operating method are reflected in behaviour over time.

Time behaviour of PLC with position control

The duration of a program cycle for a PLC is heavily influenced by the length of the PLC program. A typical value for the processing speed of a PLC is 1000 commands per millisecond. Because typical PLC programs are quite large, 10 to 20 ms often pass between the reading of the inputs and the setting of the outputs.

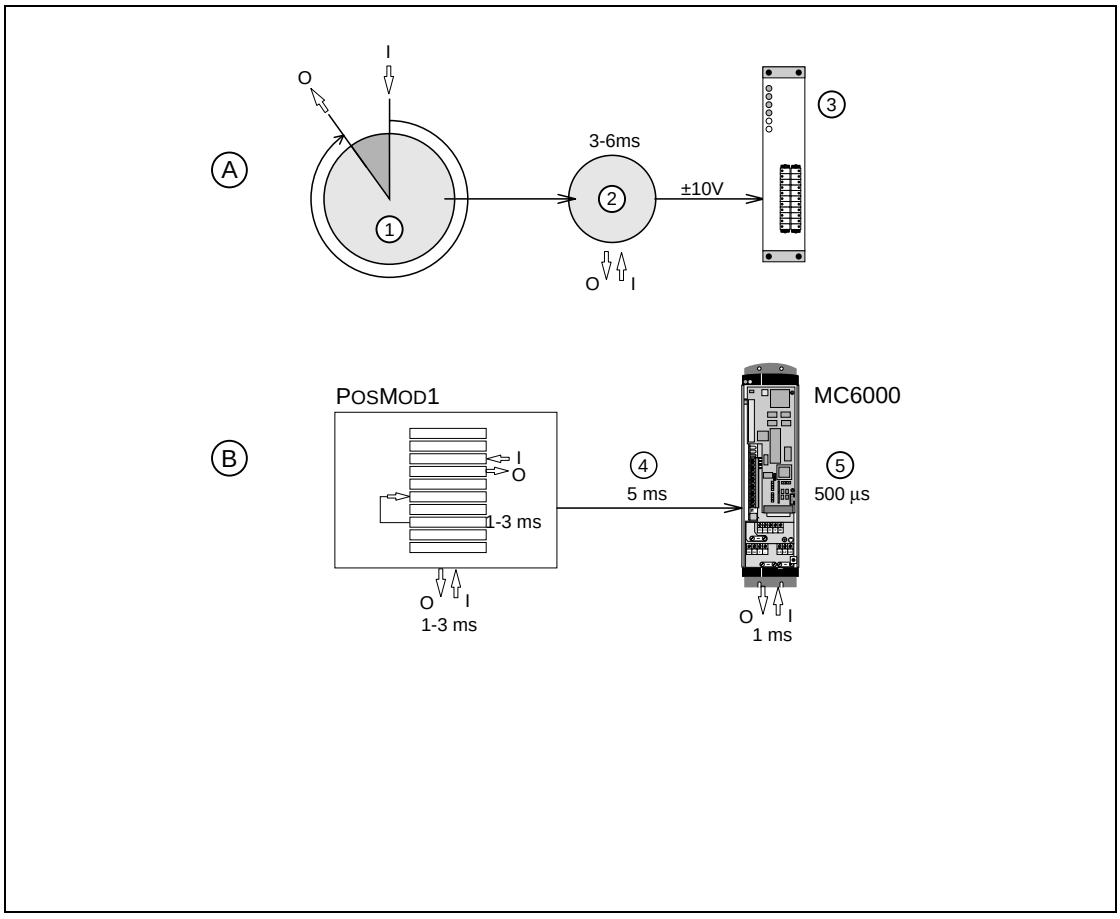
The position control in the positioning core in today's systems works at approx. 3 to 6 ms (without fine interpolation). The speed set points are sent to the servoamplifier as a ±10V analogue signal.

POSMOD1 time behaviour

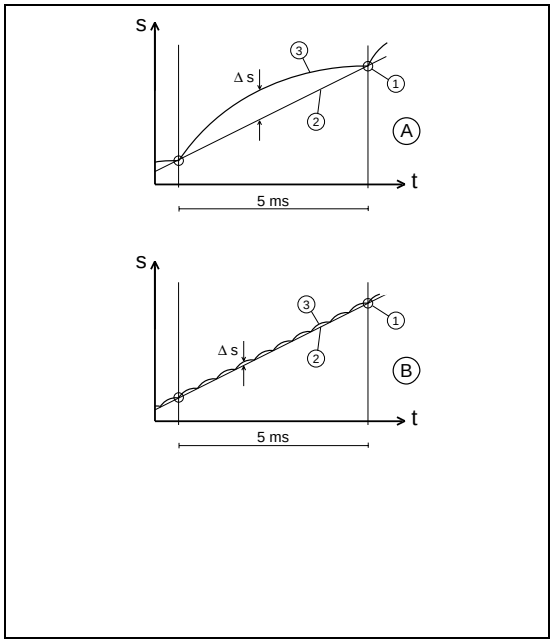
The time between reading the inputs and setting the outputs is not dependant on the length of the program. The internal sampling time is typically 1 to 3 ms, dependant on the capacity utilisation of POSMOD1. One command can be carried out in each cycle, e.g. an input can be read in. This means that POSMOD1 can react much more quickly to a change in an input, and set an output, for example.

In comparison to PLC, an individual command is carried out significantly more slowly, however. This is because POSMOD1 reserves the same sampling time for simple PLC commands as for complex positioning commands. POSMOD1 therefore can not replace PLC. However, POSMOD1 offers PLC functions so an additional PLC is unnecessary for simple applications.

The positioning unit delivers position set points in intervals of 5 ms. The set points are transferred finely interpolated to the position control circuit, which works at 500 µs Set point calculation is carried out continuously. The transfer is interrupt controlled. The position set point input therefore does not jump.



No.	Function
A	Positioning systems from PLC with position control and servoamplifier
B	Positioning systems from MC6000 with POSMOD1
1	PLC program cycle
2	Positioning core (position control)
3	Servoamplifier
4	Transfer of position set points
5	Servocontroller with fine interpolation



Fine interpolation
The s/t diagram shows that the fine interpolation clearly reduces path deviations and that the path curve is kept to more closely.

The path deviation in a sampling instant (transfer of position set point) is designated as a "lag distance" or “tracking error“. Lag distance = 0 is assumed here.

No.	Function
A	without fine interpolation
B	with 10-fold fine interpolation
1	Position set points
2	Ideal path
3	Actual path with path deviation Δs

Quantitys influencing the sampling time (typically 1 to 3 ms) are:
1. Loading from data transfer over the serial interface (e.g. cyclical status display, no longer present after commissioning)
2. Communication with CAN module (if present)
3. Communication with MC6000 (reading of actual position, set position and contouring gap)
4. Temporal phase length for the setpoint calculation routine

Maximum sampling time: 6 ms (rare peak value at maximum utilisation of POSMOD1).

Time required for:

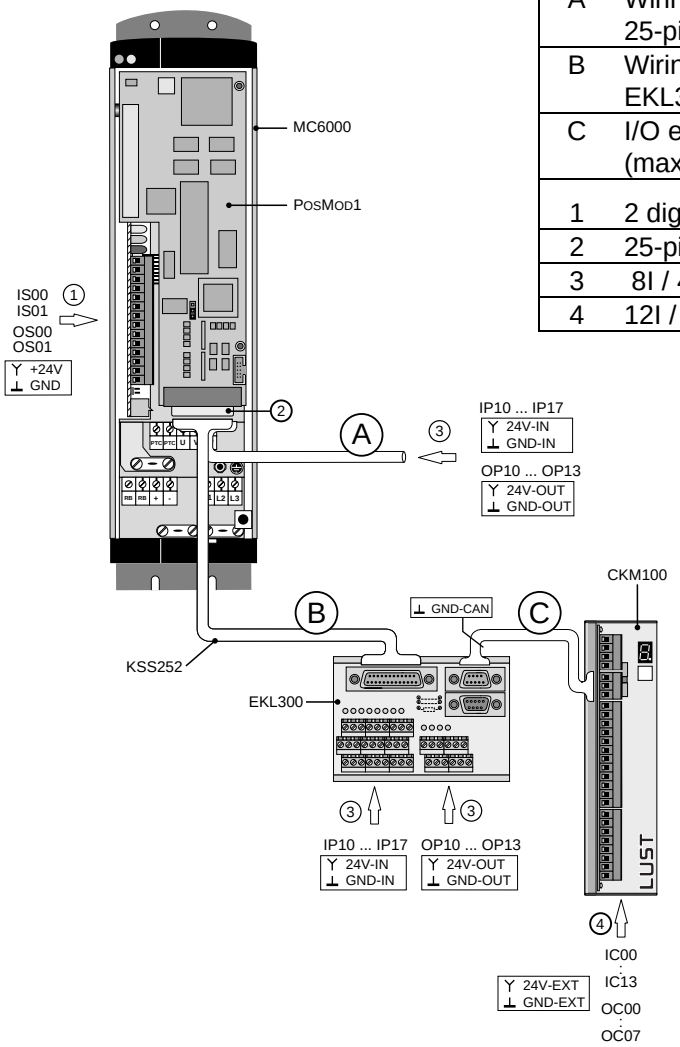
- Block-to-block execution time: 1 ... 3 ms (typical)
- POSMOD1 + MC6000 INPUTS AND OUTPUTS: 1 ... 3 ms (typical)
  Reading of inputs (JMP commands) and setting of outputs. Reactions to the functions feed hold, read-in hold, hardware limit switch and reference cam are also carried out within this period.
- SET K15 ... K24: up to 90 ms
  These SET commands require a recalculation of the acceleration parabola. Depending on the parameter settings, the execution can take up to 90 ms.
- CAN bus inputs and outputs: 7 ... 10 ms
  Reading of inputs (JMP (ICyx) ...) and setting of outputs (SET OCyx= ...) cyclical with CAN bus.
- Read status: 5 ... 10 ms
  Reading of set and actual positions and contouring gap, follows from the commands JMP (IP ...), JMP (SP ...), JMP (PW ...).
- Automatic selection and start: 20 ms
  Minimum time between selection of automatic and start.

With these inputs, the duration of a program can be set in advance. This places restrictions on the form of the input and output signals, e.g. how long a signal must remain at an input or how long it takes to set an output. The function must be checked case-by-case with the user program.

3 Electrical connections

3.1 Overview of the connections

Nr.	Function
A	Wiring of the PosMod1 I/O directly on 25-pin Sub-D socket (X16)
B	Wiring of the PosMod1 I/O over EKL300
C	I/O expansion withCKM100 (max. expansion level 22I / 14O)
1	2 dig.I / 2O (Control terminal strip MC6000)
2	25-pin Sub-D socket of PosMod1
3	8I / 4O (PosMod1)
4	12I / 8O (CKM100 CAN node module)



Explanation of the overview of connections on the next page.

Explanation of the overview of connections:

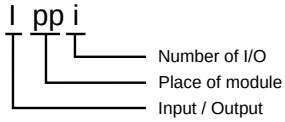
1. The graphic shows the number of inputs and outputs (I/O) which can be used with the corresponding reference potentials.
2. The CKM100 can also be supplied separately (isolated).
3. The I/O of the MC6000 are mains potential free but not optically de-coupled.
4. The I/O of the POSMOD1 and the CKM100 are isolated over optical couplers.
5. **Procedure:**
   As a rule, the I/O of POSMOD1 (GND-IN, GND-OUT) and of the CKM100 (GND-EXT) (if present) are supplied by the same power supply in the switch cabinet (wiring with the unit).

   The I/O of the MC6000 should preferably be used for local wiring and can be internally supplied.

If the ground of the servocontroller (GND) is connected to the ground of other components, these components are operated on the potential of the servocontroller (but not on the mains potential).

3.2 Designation key

The designations for the inputs and outputs of all MASTERDRIVE are found in the following designation key:



Designation input/output		Location (module)	Unit	POSMOD1 use possible
IS00...IS01, OS00...OS01		I/O standard unit	Servocontroller MC6000	yes
ISA0...ISA1		Standard unit analogue input		only ISA1
IP1x OP1x	**x= 0...7 x= 0...3**	**POSMOD1:**	**POSMOD1:**	**yes**
ICyx, OCyx	**00...07, 10...13**	**CAN bus (1 module with 12I, 8O)**	**CKM100**	**yes**
IExx OExx	xx= 00...15 xx= 00...07	I/O expansion	I/O 1 or 2	no

Above all, the inputs and outputs of POSMOD1 and, if present, the CAN module are of importance for positioning operations.

Example:
IC07 is input 7 of the CKM100 CAN module

3.3 POSMOD1 inputs

The positioning module is equipped with eight digital inputs (IP10 ... IP17).

The inputs can either be assigned defined functions, or they can be used to control the positioning process and be scanned in the positioning program.

Defined functions are, for example: automatic/manual mode switching, start, etc. Assignment is done by machine parameters. The individual functions are described in section 3.3.2, "Functions for Inputs".

3.3.1 Electrical characteristics of the POSMOD1 inputs

The inputs are isolated , and are available on the 25-pin sub-D plug connector. The +24V voltage supply for the optocoupler is to be fed in over the same plug connector. If no isolation is necessary, the internal +24V from a contact of the plug connector can also be used as an alternative.

The inputs correspond to the characteristics of the VDI 2880 (PLC process and data interface).

Electrical characteristics of the POSMOD1 inputs:

- Isolation with optocouplers
- Reverse-connect protection
- Input filter

High level:	13 ... 30,2 V	3 ... 20 mA
Low level:	-30 ... 5 V	
Operating voltage:	24 V (typical)	6 mA
Delay time:	ca. 2.6 ms	

- Reaction time for reading of POSMOD1 inputs in programs: 1 ... 6 ms

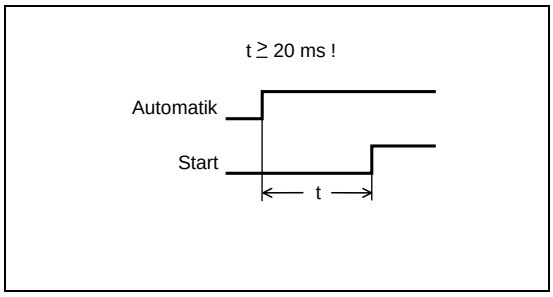
For wiring diagram see section 3.5, "Input and output wiring of the POSMOD1"

3.3.2 Functions for inputs

The inputs of the positioning module are assigned functions using the machine parameters K00 to K09 (see also chapter 5.1).
As a rule, not all functions are used. Only the functions "Automatic" and "Start" are always necessary.

Automatic: Input **IP10** or **IS00** (fixed assignment)
In automatic mode, programs stored in the positioning unit are started with the input IP11 or IS00 "Start". Automatic mode is selected by setting the "Automatic" input to H level. The number of the program to be started can either be selected with inputs (see "Program Selection" input function) or be firmly designated in the machine parameters.
Manual mode (jog mode, setup mode) is selected with L level on the "Automatic" input. If the H level is recalled during positioning, the axis is stopped immediately and automatic mode is discontinued.
The inputs "Feed hold" and "Read-in hold" (if configured) are effective in automatic mode.
After a program abort due to deselection of automatic mode, the program cannot be continued. If you wish to continue a program, the use of the read-in and/or feed hold inputs is recommended.
The function selector FIS00 = OPTN2 is to be programmed when the IS00 input is used (factory setting). The effective entry is fixed with LUPOS.

Start: Input **IP11** or **IS01** (fixed assignment)
The function of this input is dependant on the "Automatic" input. In automatic mode, the selected program is started with this input.
In manual mode (automatic = L level), a reference run is carried out when a edge change from L to H takes place at the start input.
The function selector FIS01 = OPTN2 is to be programmed when the IS01 input is to be used (factory setting). The effective entry is fixed with LUPOS.
The start signal can be retracted after 10 ms (start impulse).

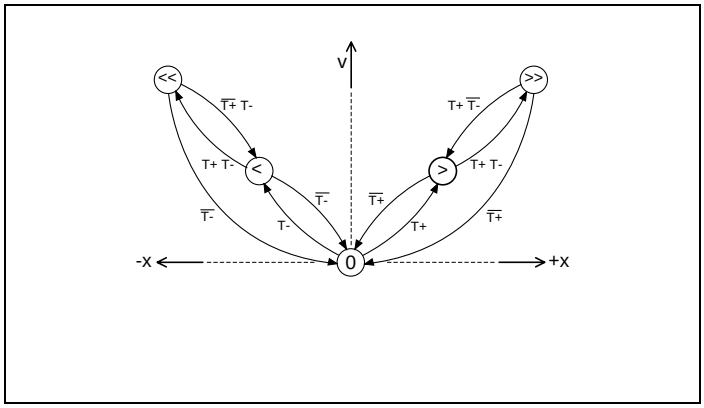


Feed hold: Input **IP1x** or **ICyx** (only in automatic mode)
The feed release (if configured) is a prerequisite for all axis movements. It allows feed movements of the axis by setting the input to H level.
If this signal is removed (L level) during an axis motion, the axis is stopped immediately with the programmed deceleration. The position control is not switched off. After restoration (H level) of this signal, the interrupted motion is continued automatically.

Read-in hold: Input **IP1x** or **ICyx** (only in automatic mode)
The read-in release (if configured) allows the processing of the blocks of a program.
If this signal is removed (L level), the program is interrupted. However any positioning motion underway is completed.
After restoration (H level) of this signal, the interrupted motion is naturally continued automatically at the point of interruption.

Jog +/-: Input **IP1x** or **ICyx** (only in manual mode)
In jog mode, the axis can be moved with these two inputs (if configured) in both directions at the creep feed or rapid feed defined in the machine parameters.

The assignment of positioning conditions to the input levels can be seen in the graphic.
Example: To move the axis in a positive direction at creep feed, use the input "Jog+". If the second input ("Jog-") is then switched on, the axis moves at rapid feed.



+x/-x	Distance pos. /neg. direction
v	Positioning speed
0	Stillstand
>	Creep feed pos. direction
>>	Rapid feed pos. direction
<	Creep feed neg. direction
<<	Rapid feed neg. direction
T+ / T-	Jog+/Jog - activated
$\overline{T+}$ / $\overline{T-}$	Jog+/Jog- input not activated

Program selection: Input **IP1x** or **ICyx**
The program to be started in automatic mode can be selected with inputs. The input terminals and the desired coding (uncoded, binary or BCD) are configured in the machine parameters.
The selection of the program to be started must be made in manual mode before switching to automatic mode! If the selected program is not available, the error message "Selected program not available!" is issued.

Table index: Input **IP1x** or **ICyx** (only in automatic mode)
With the terminals configured in the machine parameters, one can position to a position or over a relative positioning distance by adding an index. The appropriate values are stored earlier in a table.

Hardware limit switch: Input **IP1x** or **ICyx**
Two POSMOD1 inputs can be configured for assessment of one hardware limit switch each (HLS) in positive or negative direction of the axis.

If a hardware limit switch is activated, the axis motion is stopped immediately at the maximum linear braking ramp (machine parameter K19 or K20). The fault is signalled by the LUPOS user interface as well as by output OP13 or an output of the servocontroller if it has a fault signalling function.

If a limit switch is approached, the axis can be moved free in the other direction in manual mode (jog mode, manual mode through LUPOS or reference run). During the reference run, the limit switches have the function of reversing direction.

Hardware limit switches ("safety limit switches") are LOW-active, i.e. they are to be wired as openers. A break in a line therefore corresponds to an activated limit switch.

Reference cam: Input **IP17** or **IS00** (fixed assignment)
The input IP17 is intended for the connection of a reference cam and therefore equipped with special hardware. Configuration in the machine parameters is not necessary.
For application which do not require a reference cam (e.g. when using a type G3 Multiturn sensor), the IP17 input can be used as desired.

If the IS00 input is to be used to connect the reference cam, the jumper is to be inserted appropriately on POSMOD1 (see 1.1 "plan“).

3.4 POSMOD1 outputs

POSMOD1 is equipped with 4 digital outputs (OP10 ... OP13). The number of outputs can be increased with a CAN bus module (external outputs) if the application requires.

The outputs have defined functions, which are firmly assigned to the output by the machine parameters. As an alternative, the outputs can also be given to the positioning programs to be used freely.

3.4.1 Electrical characteristics of the POSMOD1 outputs

The outputs are isolated , and are available on the 25-pin sub-D plug connector. The +24V voltage supply (24V-OUT) should be supplied externally through the sub-D socket. If no isolation is necessary and the maximum load is < 200 mA, the internal +24V from a contact of the plug connector can also be used as an alternative.

In the test phase after switching on the servocontroller, all POSMOD1 outputs are set to high for a short time.

Electrical characteristics of the POSMOD1 outputs:

- Isolation with optocouplers
- Short circuit-proof
- Thermally protected
- Error messages for:
  - Short circuit against GND (current limitation)
  - Thermal overload
  - Undervoltage (< 9 V)

Operating voltage:	24 V	(typical)
Power loss	250 mW	(max.)
ohm load	500 mA	
inductive load	12 W	
Lamp load:	250mA	
Capacitive load	2 µF	

- Completion time for setting of POSMOD1 outputs in programs: 1 ... 6 ms

For wiring diagram see section 3.5, "Input and output wiring of the POSMOD1"

3.4.2 Functions for outputs

The following describes the functions which can be assigned to the POSMOD1 outputs. One can determine with machine parameter K05 whether an output is carrying out its defined function or can be freely responded to by the positioning program (factory setting: defined function).

Program End: output **OP10** (fixed assignment)
The end of program processing is signalled by this output if automatic mode is selected:

H level: No program is being carried out or the end of the program has been reached.
L level: Program processing underway

Axis in position: output **OP11** (fixed assignment)
If selected, this output indicates if the axis is located within the position window.

H level: Axis within the position window
L level: Axis outside of the position window

The output is set to H level as long as the control system is not activated. No set point has been submitted to the servocontroller. The set position follows the actual position.

Home position defined: output **OP12** (fixed assignment)
If selected, this output signals whether a reference point has been defined. This happens either after a successfully completed reference run or after transfer of the current position as a reference point.

H level: Home position defined
L level: Home position is not (yet) defined

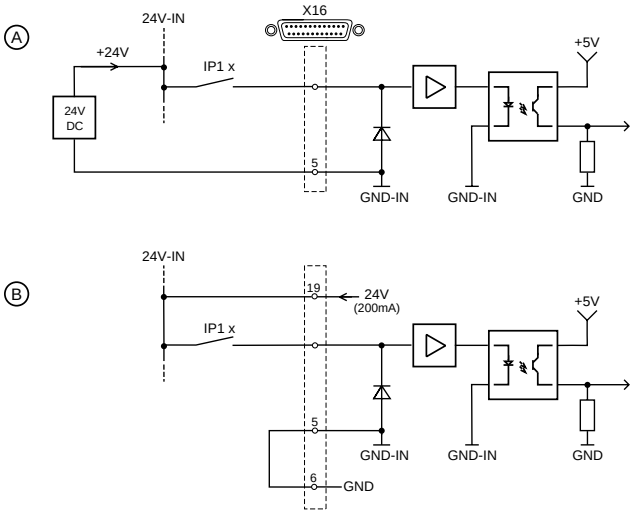
Fault signal: output **OP13** (fixed assignment)
If selected, this output signals an error recognised by the positioning unit. Resetting the error is done either over the serial interface or through measures which lead to a solution of the error.

H level: Malfunction of positioning unit
L level: No malfunction of positioning unit

3.5 Input and output wiring of the PosMod1

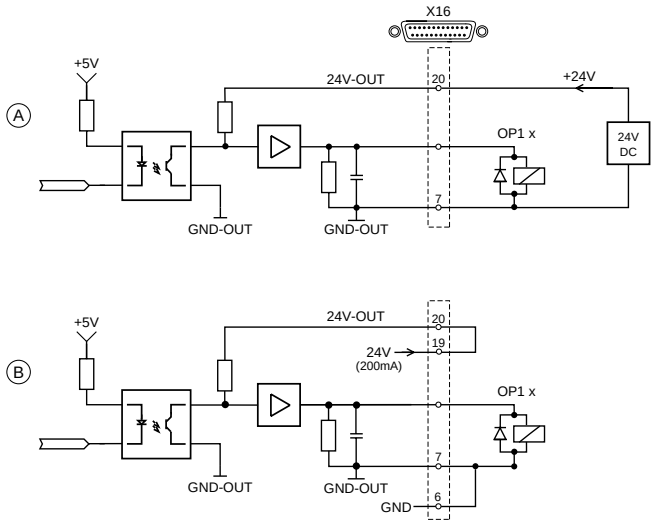
Two wiring possibilities:	
A	Isolated with external power supply
B	Not isolated, supply from servo (+24V), total load capacity: 200 mA!

IP10 inputs... IP17:



X16 connector connection assignment, see section 3.10.

OP10 outputs ... OP13:



Isolation strongly recommended, if

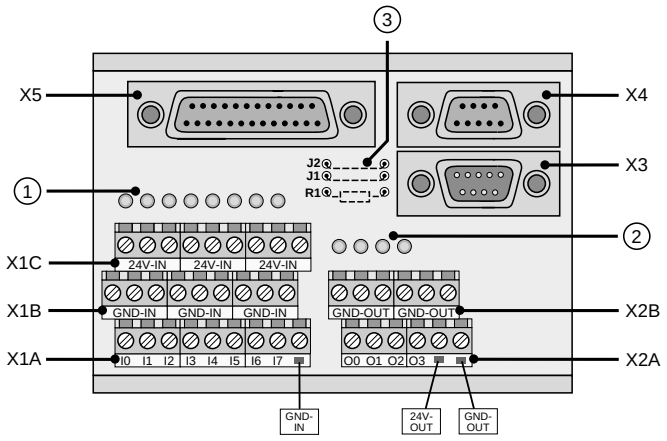
- wiring runs over great distance or
- several control units are connected.

3.6 EKL300 terminal module

The EKL300 is available as an accessory and is intended for installation on a PE rail in the cabinet. With the terminal module, the wiring of PosMod1 inputs and outputs can be done in the cabinet. The level of an input or output is displayed with LED's.

The EKL300 (X5) is coupled to PosMod1 with a 25-pin connection cable (order code KSS252). Inputs IP10 to IP17 can be wired to the X1 terminals, outputs OP10 to OP13 to X2.

The terminal module is constructed in the three-wire technique. With the three lines – +24V, signal and ground - initiators (e.g. proximity switches) can be connected conveniently.



No.	Function	No.	Function
1	LED's for 8 inputs	X2A	Output terminals
2	LED's for 4 outputs	X2B	Ground for outputs (GND-OUT)
3	CAN bus attachment possibilities	X3	CAN bus input
X1A	Input terminals	X4	CAN bus output
X1B	Ground for inputs (GND-IN)	X5	Coupling on PosMod1
X1C	+24V for inputs (+24V-IN)		

Terminal designations: (on the EKL300, jointly for PosMod1 and I/O module)

EKL300	PosMod1 (8E/4A)	E/A-Module 1 (8E/4A)
I0	IP10	IE00
:	:	:
I7	IP17	IE07
O0	OP10	OE00
:	:	:
O3	OP13	OE03

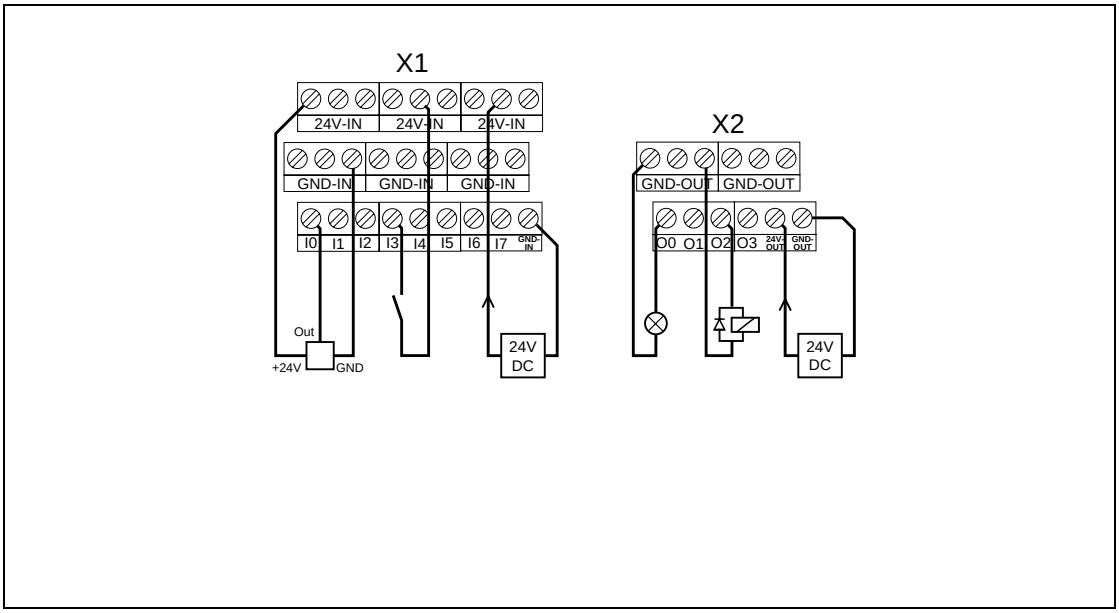
P1 = PosMod1
E = I/O module

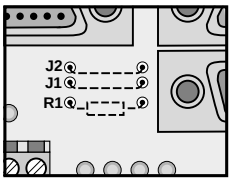
(compare 3.2 "Designation Key")

Power supply for inputs and outputs

Supply +24V externally for isolation. Inputs and outputs can be supplied isolated from one another. Supply can take place over an X1C terminal for inputs and over X2A-5 for the outputs.

Basic connection:





EKL300 detail

The X3 and X4 connectors as well as the EKL300 jumpers are only necessary if the number of inputs and outputs has been expanded with the CKM100 CAN module (see chapter 3.8).

The connection assignment of the X5 corresponds to the X16 connector of PosMod1 (see chapter 3.9).

3.7 Servocontroller inputs and outputs

The inputs and outputs of the MC6000 basic unit can be used with PosMod1:

a) with functions of the MC6000 servocontroller or

b) with functions of the PosMod1 (factory setting).

Basic principle: The inputs and outputs are assigned defined functions with the function selectors (see MC6000 Operating Instructions).

a) Functions of the MC6000 servocontroller

Inputs (ISA0, ISA1, IS00 u. IS01):

Function selector setting	Function of the input
OFF	none
/STOP	Activate quickstop (low-active)
E-EXT	external error
OPTN1	available for module in optional slot 1
OPTN2	available for PosMod1, *see b)*
SCALE	Torque limitation 0 ... 100% (only ISA1 input)

Outputs (OS00 & OS01):

The outputs can be assigned all the functions described in the servocontroller Instruction Manual except for the function "Limit" - limiting value reached.

b) Functions of the PosMod1

Inputs (ISA0, ISA1, IS00 u. IS01):

Function selector setting	Function of the input	
OPTN2	available for PosMod1 IS00: Automatic IS01: Start ISA0: no function ISA1: Override	**Factory setting** (when switching on for the first time after attaching PosMod1)

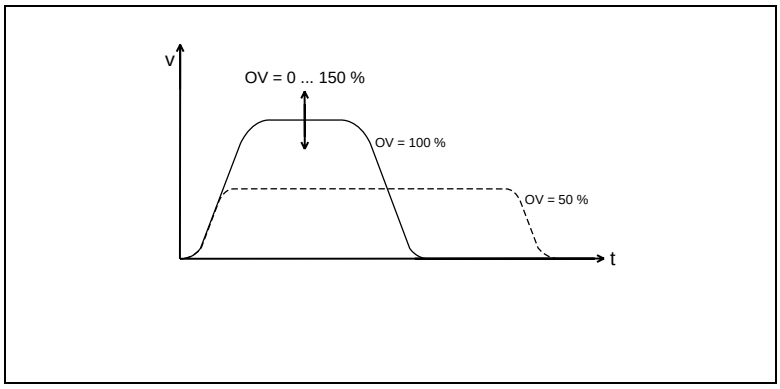
Outputs (OS00 & OS01):

Function selector setting	Function of the output	
OPTN2	OS00 and OS01 free programming in application program	**Factory setting** (when switching on for the first time after attaching PosMod1)

Special functions of POSMOD1:

Override: **ISA1** input (factory setting)

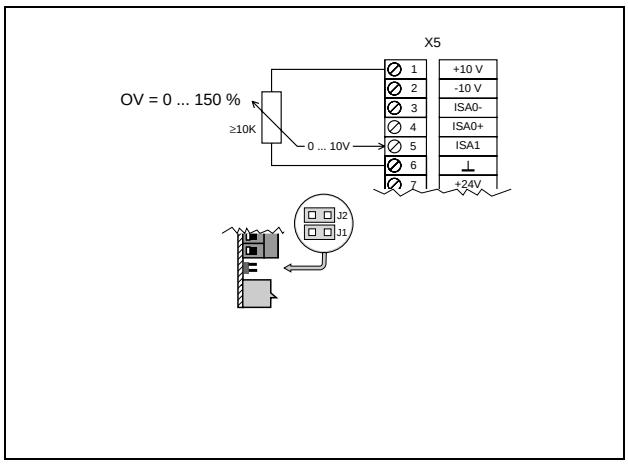
By using override (OV), the positioning speed can be adjusted in a range from 0 to 150%. This is useful above all when setting a new axis program, to carry out a test at sharply reduced speed (e.g. 40%). This special function can only be carried out by the ISA1 analogue input.



Wiring:

Jumpers J1 and J2 under the control terminal strip

	Pos.	Function
J1		none (standard)
		not allowed
J2		Override 0 ... 10 V
		Override 0 ... 20 mA



Necessary configuration for MC6000:
FISA1 = OPTN2 (_CONF)
(factory setting)

If the override is not needed,

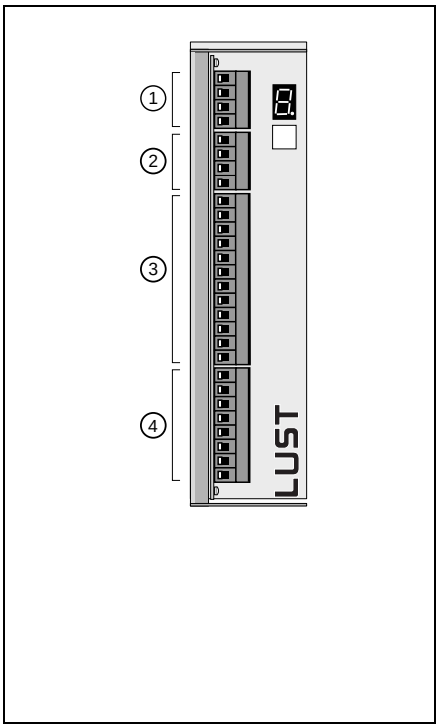
a) the ISA1 input must be assigned another function or
b) the override must be switched off at the beginning of the axis program with the command SET OV=0.

Otherwise the axis will not move because an override = 0% is recognised when the ISA1 input is unconnected!

Note:
The override should only be changed while the axis is standing still. Otherwise erratic changes in the speed set point can cause the drive to buck!

3.8 I/O expansion with CKM100

It is possible to increase the number of inputs and outputs with the CKM100 CAN bus interface with 12I/8O.

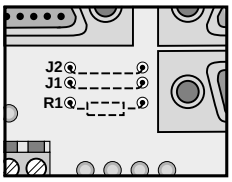


No.	Function
1	24V power supply
2	CAN connection (bus module input)
3	12 inputs
4	8 outputs

On PosMod1, the CAN bus signals contact the 25-pin sub-D connector (X16). For EMC engineering reasons, a paired, screened cable is required for the CAN bus connection (included with the CKM100).

Power supply for the CKM100

The +24V power supply for the CAN bus can either be applied on the CKM100 module or on the EKL300.



EKL300 detail

For supply through the EKL300, equip both solder bridges J1 and J2. This bridges the 24V CAN and GND-CAN lines of the CAN bus to 24V-OUT and GND-OUT.

The 24V-OUT voltage must then be applied to the EKL300. The power supply from the servocontroller is not designed for this purpose.

Line termination:
A CAN bus must always be terminated with 120 Ω on the beginning and end.

The EKL300 acts as a line end, the other end is the CKM100 bus module. The resistance R1=120 Ω is to be switched on at the terminal module (equip or jumper).

EKL300 X3, X4	Allocation
2	CAN configuration
3	GND-CAN
7	CAN+
9	24V-CAN

Electrical characteristics of the CKM100 inputs:

- Isolation with optocouplers
- Reverse-connect protection
- Input filter

High level:	13 ... 30,2 V	3 ... 20 mA
Low level:	-30 ... 5 V	
Operating voltage:	24 V (typical)	6 mA
Delay time:	ca. 2.6 ms	

- Reaction time for reading of CKM100 inputs in programs: 7 ... 12 ms

Electrical characteristics of the CKM100 outputs:

- Isolation with optocouplers
- Short circuit-proof
- Thermally protected
- Error message throutgh status with:
  - Short circuit against GND
  - Thermal overload
  - Undervoltage (< 9 V)

Operating voltage:	24 V	(typical)
Power loss	250 mW	(max.)
ohm load	500 mA	
inductive load	12 W	
Lamp load:	250mA	
Capacitive load	2 µF	

- Completion time for setting of CKM100 outputs in programs: 7 ... 12 ms

Notes on CKM100 configuration:
The CKM100 must be assigned an address after installation (see section 3.9.4 "Note adresse of the CKM100") which corresponds to the address in the K60 machine parameters.

3.9 CKM100 operations

The module is equipped with a 7-segment display and a button for displaying operating conditions and setting parameters.

3.9.1 Status display

The display normally contains one numeral which indicates the status of the module.

The following status displays are defined:

0 Initialisation phase
1 Standby
2 Node module active in event mode
3 Node module active in poll mode

3.9.2 Resetting the CKM100 node module

This initiates a restart of the node module software.

Operation:
Press button briefly 3 times until an "r" appears in the display. The hold the button down until the display flashes. After repeated pressing, the reset is activated. As long as reset is active, three horizontal segments are displayed.

3.9.3 Parameter display and settings

The parameters

- Baud rate
- node address

can be displayed and set.

Display of parameters:
After repeated brief pushing of the button, the parameters can be selected. These code letters appear in sequence in the display:

- "A" node address
- "B" Baud rate
- "r" Reset

If no more buttons are pushed for approximately 3 seconds while the display of the desired parameter code letter is active, then the display alternates between showing the value of this parameter and the parameter code letter.

Pushing the button again switches to the next parameter. If no more buttons are pushed for approximately 10 seconds, the CKM100 returns to "Status display" mode.

Setting parameters:

- Repeatedly press button briefly until the desired parameter code letter appears in the display.
- Release button briefly and then hold down until the code letter flashes.
- The parameters can finally be set by pressing the button repeatedly.
- If no more buttons are pushed for approximately 5 seconds, the CKM100 returns to "Status display" mode and programming is completed.

The new values are permanently stored internally. They remain even when the power supply is switched off. The range of the values is already checked during entry, so that setting invalid values is impossible.

IMPORTANT! The changed values do not take effect until the power supply is switched on again for the first time or until after reset has been activated.

3.9.4 Node address of the CKM100

Setting the node address:
The node address can be set as follows in the range from 1 ... 31.

Attention! The SELECAN protocol only allows an address range from 1 ... 29!

The hold the button down until the code letter "A" (address) is displayed.
Release button briefly and then press again until the code letter "A" flashes.

Release button.

Display alternately:
↕ Code letter for higher digit H (tens column) blinking
↕ Node address of the tens column

Repeatedly press button until the desired value appears.

If no more buttons are pushed for approximately 5 seconds, the following appears automatically switched further to the lower digit:

Displayed alternately:
↕ Code letter for lower digit L (ones column) blinking
↕ Node address of the ones column

Repeatedly press button until the desired value appears.

If no more buttons are pushed for approximately 5 seconds, the module status is displayed again and programming is completed.

Display of node addresses:
The hold the button down until the code letter "A" (address) is displayed.
Release button.

Displayed alternately:
↕ Code letter A
↕ Node address of the tens column
↕ Node address of the ones column

If a button is pushed again, the baud rate display (code letter "b") is switched on.

If no more buttons are pushed for approximately 10 seconds, the module status is displayed.

3.9.5 Baud rate of the CKM100

Display of baud rate:
The transmission rate on the CAN bus is displayed as follows:

The hold the button down until the code letter "b" (baud rate) is displayed.
Release button.

Displayed alternately:
↕ Code letter b
↕ Baud rate code number 1 ... 6

Baud rate	Code number
20 kBit/s	1
50 kBit/s	2
100 kBit/s	3
125 kBit/s	4
250 kBit/s	5
500 kBit/s	6

The baud rate is set and fixed at 500 kBit/s (display b6).

If a button is pushed again briefly, the reset function display (code letter "r") is switched on.

If no more buttons are pushed for approximately 10 seconds, the module status is displayed again.

3.9.6 CKM100 error display

If an error arises in CKM100, it is announced to the servocontroller in the status telegram.

In addition, the error is displayed on the seven segment display of the CKM100 alternating between the following:

↕ Code letter F
↕ Error number of the tens column
↕ Error number of the ones column

Error No.	Explanation	Comment
CKM100 function errors:		
01	Short circuit in CKM100	
02	Error in local CAN of the CKM100	Check ribbon cable.
03	Timeout between master and CKM100 (global CAN)	Ckm100 was not addressed by the master during Start-up repetition time". CKM100 goes into standby status.
CAN communication errors in global CAN		
80	Bus off error	CKM100 goes into standby mode.
81	Error, passive	Check ribbon cable.

If several errors arise at the same time, they are displayed according to the following priorities:

1. Bus off error (highest priority)
2. Error, passive
3. Function error (lowest priority)

If several function errors arise at the same time, the error arising first is displayed first.

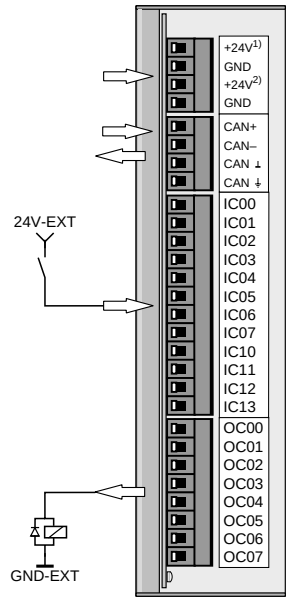
Each error is displayed at least once. This means that the 'F-DezStelle-Einerstelle' cycle is run through at least once.

The error are always displayed with a two-figure display.

The error display can be deleted in the following manner:

- by pressing the button
- by activating the CKM100 with a start communication telegram
- if the CKM100 successfully sends a status telegram back to the master while in active status (event or poll mode) on command of the master.

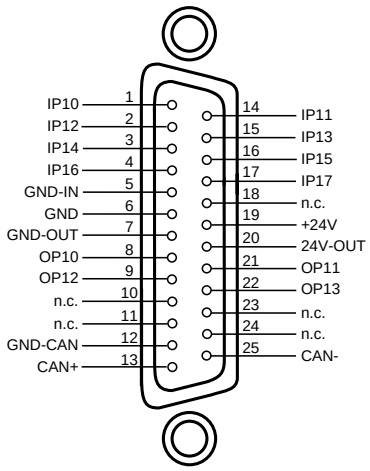
CKM100 terminal assignment:



1) 24V power supply for CPU and logic
2) 24V power supply for I/O levels

3.10 Connection assignment (X16)

Pin assignment of the X16 connector on the POSMOD1 (25 pin sub-D socket):



Wiring to the unit can be carried out either with an appropriate connecting cable or on the EKL300 external terminal module.

4 LuPos user interface

4.1 Installing and calling up LuPos

LuPos is the PC program for programming and setting parameters for the PosMod1 position control system. The user interface is delivered on a diskette which can be read in directly. By entering LUPOS, the program can be started immediately by the diskette.

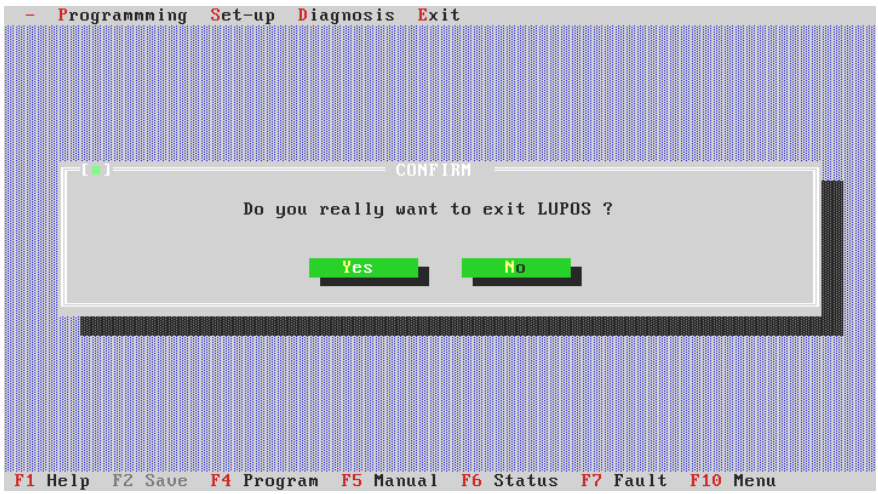
To install on the hard drive, copy the file and directories of the diskette into a directory of the hard drive. This can be done with the Microsoft Windows™ file manger, for example, or in DOS. One should insure that the directory structure of the diskette is maintained.

The procedure when using DOS is described in the following. It is assumed that the diskette is located in drive A: and that the hard drive is designated with C: The entries should be changed correspondingly if necessary.

1. Creation of a directory for the user interface:
   MD C:\LUPOS <Return>
2. Copying the files and directories:
   XCOPY A: C:\LUPOS /s/e <Return>
3. Copying the LUPOS.BAT start file in the root directory C:\ :
   COPY A:\LUPOS.BAT C:\ <Return>
4. Setting the language with the Editor:
   EDIT C:\INFOS.DAT <Return>
   L:D (D=German, GB=English)
   :

The batch file to start the user interface is called up with LUPOS <Return>. After copying, it is located in the root directory of the hard drive. The LUPOS.BAT file assumes that the LUPOS directory is a directory of the root directory C:\. If this is not the case, LUPOS.BAT must be adjusted appropriately.

Please start the program now: LUPOS <Return>



LuPos displays the information window, which contains the version number. The information window is closed by entering <Return>.

4.2 Layout and operations of LuPos

The LuPos user interface is designed according to the SAA standard for user interfaces. It can be operated with the keyboard as well as with the mouse.

The program has a menu bar in the top line of the screen. The bottom line displays which functions can be reached with function keys. The area between the top and bottom line can be seen as the working area for the menu windows.

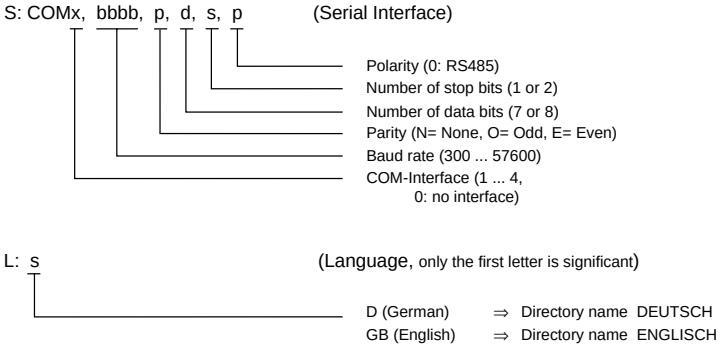
The pull-down menus can be activated by clicking with the mouse or by pressing <Alt> and the corresponding highlighted letter. One can scroll through the menus with the cursor key. Menu points are selected with the <Return> key or directly with the mouse.

In accordance with the SAA standard, one can switch to the right with the tab key and to the left with <Shift> + <Tab>. Using the mouse is more convenient, however. As an alternative, the procedure can be carried out with key combinations consisting of the <Alt> key and the highlighted letter.

Actions can be undone or aborted with the <ESC> key. The functions of the function keys and the <ESC> key are described exactly in section 4.6 "Function Key Assignment".

4.3 Configuration of LuPos

The user interface is configured with the INFOS.DAT file. This can then be edited with a text editor (e.g. EDIT in the DOS operating system). The baud rate of the serial interface of the PC can be set here. The standard setting is 9600 baud, as with the servocontroller.



Factory setting: S:COM1,9600,7,1,1 Enter COM2 when the mouse is operated on COM1!
L:D

Insure that the baud rate of the PC corresponds to the baud rate of the servocontroller. Otherwise, no connection can be made. In this case, LuPos announces the error that no axis is present.

For information on setting the baud rate of the servo, see the Instruction Manual for the MC6000 servocontroller.

4.4 Menu structure

The user interface has a structure consisting of five menus which are explained individually in the following:

-	Programming	Setup	Diagnosis	End
• Information	• Load program • Save program • Save p. directly • Save p. as • Print program	• Reference run • Manual mode	• Status display • Error report	• End
	• Machine parameters • Variables • Table positions			
	• Save data			

4.4.1 Information menu

This menu window is also displayed when LUPOS is called up and contains the version number of the software.

4.4.2 Programming menu

Positioning programs are created and managed in the programming menu. The positioning axis (axes) is (are) configured here with machine parameters. Table positions and variables are managed here as well. Axis data is written into the non-volatile axis memory with the last point in the menu.

After selecting a menu point, a window with three or four buttons appears: disk, axis, (new) and abort. Here, for example, one indicates whether the program should be loaded by the PC or by the axis and whether it should be newly created or whether the operation should be aborted. If one selects "Disk", one finds a list of the available files in the next window from which the desired program can be selected. If the program is to be loaded by the axis, the number of the axis is requested in the next window. If it is a single axis positioning application, i.e. there is not more than one servodrive meshed on the PC, confirm "0" as the axis number there.

Load program <F4>
After the procedure described above, a program saved on a diskette, hard drive, or on the axis is in the program editor available for processing, or the program editor is opened for creation of a new program. The editor and its functions are described in more detail 4.5 "Program Editor". Programming and the instruction set are dealt with in chapters 7 and 8.

Save program <F2>
With this menu point, an available program can be saved on a diskette, hard drive or axis. The last-used source or target location is always the target location (file name) for storage.

Saving a program directly
With this function, one can store a program without error monitoring. This is useful for the storage of flawed programs.

Save program as **<ALT> + <F4>**

This menu point offers the additional possibility to specify a different target location, i.e. a different file name or a different axis number. The axis number corresponds to the interface address from 0 ... 30. When only one axis is connected, use axis number = 0.

Print program **<ALT> + <F5>**

This point allows one to print the currently available program on any standard printer which is connected.

Machine parameters

The positioning axes are configured by using the machine parameters (K00 ... Kxx). The creation and management of the machine parameters is in sections of several on-screen pages, between which one can switch with the <Display↑>- and <Display↓> keys. The individual sections are:

- Configuration of the inputs and outputs
- Positioning parameters
- CAN bus configuration
- Reference run configuration

Four options are available on each on-screen page: "Print", "Save", "End" and "Abort". They should be self-explanatory. The description necessary for configuring the axis is located in chapter 5 "Machine Parameters".

Variables and **table positions**

With these two menu points, sets of variables and table positions can be created and managed. A set of variables consists of 100 variable (H00 ... H99), a table of 16 positions (T00 ... T15).

Store data

Programs, machine parameters, variables (H50 to H99) and table positions are referred to collectively as "axis data". If one transfers axis data to an axis using the other menu points, this data is first saved in the working memory (RAM) of the axis.

The "Store data" option must always be selected so that the data newly transferred to the axis is not lost in a power outage.

The data is read from the RAM of the axis and read into non-volatile memory (Flash-EPROM).

4.4.3 Setup menu

This menu supports the setup of the axis. After selecting a menu point, an axis number is entered (for non-meshed axes, use axis number = "0").

Reference run

With the first menu point, the axis moves to the home position according to the conditions set in the machine parameters. This creates the absolute position reference in relation to the entire axis. This procedure is started with the "Start" option and can be stopped with "Stop" if necessary. It is also possible to change axes directly with "Axis". One can leave the menu with "End".

The current status of the axis is displayed at the same time. The meanings of the various indications are described in section 4.4.4 "Diagnosis menu" under the heading "Status display".

Manual mode **<F5>**

Manual mode is used to move the axis "by hand", i.e. not in automatic mode, and also to give it other control commands (compare 2.1 "Operating Modes"). This can be done conveniently over the serial interface of the servocontroller. This operating mode is best used primarily to list and modify certain machine parameters as well as to search for errors in application programs.

In the "Speed" field, three speeds can be set: Creep feed or rapid feed as well a speed which can be set as desired. The axis can then be moved at this speed in jog mode by pressing the "+" and "-" keys. This corresponds to jog mode with digital inputs, however it is not possible to switch from creep feed to rapid feed by press both keys simultaneously. Creep feed and rapid feed are determined by the machine parameters K25 and K26.

As an alternative to jog mode, the axis can also be moved absolutely or relatively at the selected speed in the "Positioning" field. After a position or a distance is entered, positioning can be started with "Start" and stopped with "Stop".

With the "Instruction" field, it is possible to send any command of the instruction set to the axis and have it carried out in steps. Jump commands and subprogram call-ups obviously cannot be entered as they can only be used in programs. A memory bank for the last used instruction is available which can be retrieved with the ↓ key.

The commands for program and data management can also be entered as instructions and sent to the axis. The array of possibilities can be seen in the description of the commands for program and data management in chapter 9.

If one calls up the help function while the "Instruction" field is active, the instruction set with the commands for program and data management becomes available as a special feature. By pressing <RETURN>, one can switch to the instruction set (and back).

Options are available in the neighbouring manual mode menu with which another axis can be directly selected, outputs can be set and the menu can be closed. During a reference run, the current status of the axis is constantly displayed in the lower part of the window (contouring gap displayed only in increments here).

Use of the manual mode menu is especially important for:

switch off control system: STOP 0

e.g. to set servocontroller parameters. Switching off the control system switches off the current to the motor. This is not allowed in all applications without further measures (e.g. in lifting applications).

The user must insure that the unit is not damaged and that persons are not injured!

switch back on with: STOP B or STOP M.

4.4.4 Diagnosis menu

LUPOS offers the diagnosis menu to monitor the safe functioning of axes and for any necessary error search in the positioning program.

Status display <F6>

This window contains information on the current status of the axis. This information is very important to determine if a application program is being carried out in the desired manner by the axis. If this is not the case, the cause can be determined using this information.

The following information is displayed:

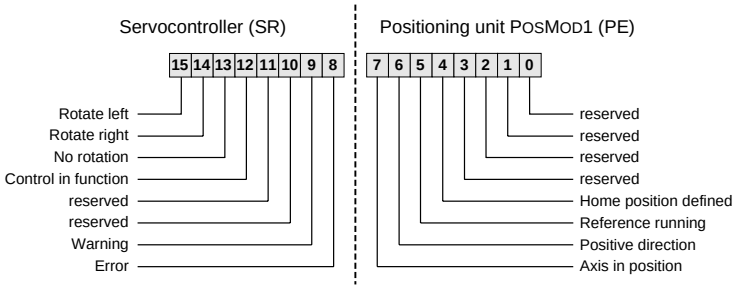
- Program number and block number
- Actual and set position (in the programmed distance unit, see Machine Parameters) **Attention!** Actual value overflow at ± 2.147.482.879 increments: must not be exceeded
- Contouring gap (in the increments of the sensor and the distance unit)
- Override (in percent)
- Status of the local inputs and outputs
- Status of the limit switch (0= limit switch not reached)
- Bits status of the servocontroller (SC) and the positioning unit (PU)

Some options are available with which further information can be called up. The status of all present CAN inputs and outputs can be displayed with "CAN-IO". The representation is carried out by module and by byte, i.e. bit 0 of the CAN node module (IC00 or OC00) is on the upper right.

With the "Variables" and "Markers" options up to 8 variables or markers each can be selected and represented. The windows can be moved with the cursor keys after pressing <CTRL> + <F5>.
When the desired position has been reached, the window can be fixed there with <RETURN>. The window active at the time (recognisable by the double outline) can be closed with <ALT> + <F3>.

The status display also allows one to switch directly to a different axis with the "Axis" option. One leaves this menu, like all others, by clicking the "End" option.

Control status:



The status display is shown in a similar but somewhat reduced form in the "Setup" menu under "Reference run" and "Manual mode".

Error report **<F7>**

The error report screen contains a list of errors which occurred last (after LUPOS has been started). The errors are described in a plain text message. This makes it possible to find any systematic errors, for example.

The software of POSMOD1 does not have an error cache which stores the errors after the servocontroller has been switched off. This is not necessary because every error causes the axis to switch off. In this case, the program and the block number, which provide important information for the error search, are indicated.

4.4.5 Menu exit

This menu is selected to exit LuPos. If axis data is to be changed, a security query follows asking whether this data is to be stored in the axis (see Securing Data).

4.5 Program editor

Positioning programs can be created and processed with the program editor, which is opened after loading a program. Several functions are available in the editor as function keys. Other functions cannot be activated until the menu is selected with <F10>; e.g. printing a program.

The program can be saved with <F2> or <ALT> + <F4>. The editor can then be closed with <ALT> + <F3>.

Caution!
If one does not store first, the program is lost when one exits the editor.

If the <RETURN> key is pressed at the end of a line, the editor checks the line for validity (command syntax). If a syntax error is recognised, an error message is displayed; e.g. "Unauthorised instruction" or "Unauthorised character". Check the spelling of the last line. The cursor usually shows the faulty section.

Note:

- Other text editors (e.g. EDIT from MS DOS) can be used to edit programs. However it cannot be guarantied that these offer the necessary characters and represent correctly. Other editors cannot carry out the syntax check either. Using the LUPOS editor is therefore recommended.
- The program is not case-sensitive.
- Place a space between block numbers, commands and operands (no tabs).
- Commentary is only secured when stored on the hard disc or diskette. It is not stored in the axis to save storage space in POSMOD1.

The program editor of LUPOS is command-compatible with the Turbo C Editor from Borland. The table contains the most important functions of the editor.

Block operations	
<STRG> + <K> , <B>	Mark beginning of block (press <STRG> and <K> simultaneously, release, then press <B>).
<STRG> + <K> , <K>	Mark end of block (copy block in storage).
<STRG> + <K> , <C>	Copy block (insert)
<STRG> + <K> , <Y>	Delete block.
Search / replace	
<STRG> + <Q> , <F>	Search
<STRG> + <Q> , <A>	Search and replace
<STRG> + <L> , Cursor →	Repeat search and replace
Other	
<STRG> + <N>	Insert empty line above cursor position.
<STRG> + <Y>	Delete line
<STRG> + <Q> , <Y>	Delete up to end of line.
<STRG> + <T>	Delete rest of word (after cursor position).
<STRG> + <A>	Word to the right
<STRG> + <F>	Word to the right
<STRG> + <Screen↑>	Jump to beginning of text.
<STRG> + <Screen↓>	Jump to end of text.
<ALT> + <F3>	Exit editor (save first!)

4.6 Function key assignment

The function keys in LUPOS are assigned the following functions:

Key(s)	**Function**
<ESC>	Abort, undo
<F1>	Help
<F2>	Save program
<F4>	Load program
<F5>	Manual mode
<F6>	Status display
<F7>	Error report
<ALT> + <F3>	Close window
<F4>	Save program as
<F5>	Print program
<CTRL> + <F5>	Move status display window

Note:
Not all functions are always available. Availability depends, for example, on whether a program is loaded or not. An exception is the <F1> key. Help is naturally always available.

Function of the ESC key
In keeping with the SAA standard, the <ESC> key is used for the function "Abort" (without saving). To avoid losing numerical values by pressing the <ESC> key unintentionally, the function has been extended slightly for all numerical entry fields. Here, the <ESC> key has the effect that the old numerical value is restored. This is useful above all for machine parameter entries. Aborting the machine parameter dialogue is possible by selecting "End" when the following security query, "Save programmed data?", is negated.

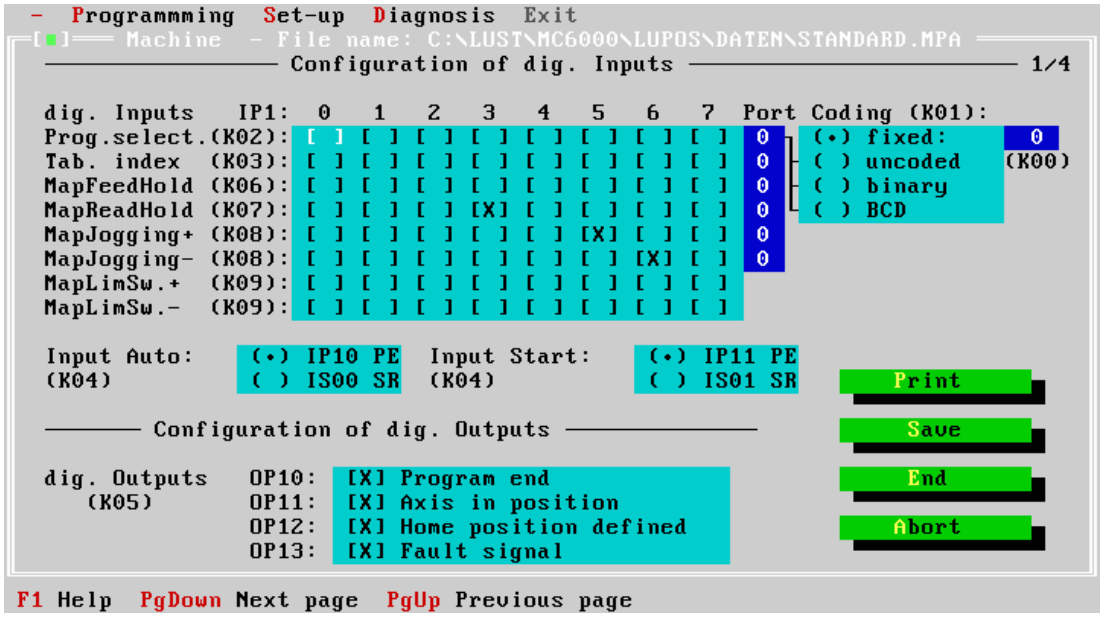
5 Machine parameters

Machine parameter programming is carried out menu-controlled with the user software of the positioning unit in the "Programming" menu.
All parameters with their meanings and value ranges are then described.

Important recommendation:
Print out the machine parameters and add commentary after setup. This makes the parameter settings easier to understand, even at a later date (e.g. distance resolution K10, K11).

Note:
If no values for the parameters are entered, the setting stored in PosMod1 are used after the transfer (at the first transfer: factory setting).

5.1 Configuration of the inputs and outputs



Operation note:

- A function (x) is selected with the space bar or the left mouse button.
- One switches forward with the tab key <→> and back with <SHIFT> + <→>.
- "Port" refers to 8 inputs:

Port	Input			Module
0	IP10	...	IP17	PosMod1:
1	IC00	...	IC07	CKM100
2	IC10	...	IC13	CKM100

K00 Program number source

Value range: 0 ... 99

In this machine parameter, one indicates the number of the program to be started in automatic mode. The parameter is only effective if K01 = "Fixed", i.e. only if a particular program is to be called up.

K01 Program number coding

This parameter determines whether the program indicated in K00 or a program selected independently of inputs is started in automatic mode.
If program selection is to be done by inputs (not "fixed"), the inputs are to be configured correspondingly with K02. The selection of the program must be made in manual mode before switching to automatic mode. The inputs for program selection can also be used in the program or be assigned a function which is only effective in automatic mode (table index, feed and read-in hold).

Setting		Function	Inputs max.	Programs
0	fixed	Program from K00	-	0...99
1	uncoded	Program selection uncoded (one input per program)	8 [1)]	0...7
2	binary	Binary program selection coding	7 [2)]	0...99
3	BCD	BCD program selection coding	8 [2)]	0...99

Note:

- The value of K00 has no influence with the settings "Uncoded", "Binary" and "BCD".
- The two columns on the right indicate the maximum number of inputs which can be used and the programs which can be selected with them.

1) If several inputs are used simultaneously, the one with the highest value is always effective. The input with the lowest value selects program 0. If no input is set, error 16 "selected program not available" is caused.

2) The input with the highest number (according to the designation key) determines the bit with the highest value. If no valid BCD combination is set, error 16 "selected program not available" is caused.

K02 Program selection configuration

This parameter determines which inputs the program number is determined from. The local inputs IP10 ... IP17 or inputs of a CAN input module can be selected.

The parameter K02 is only effective if K01 = "Uncoded", "Binary" or "BCD" (see description of K01).

Note:
When printing or using a user interface different than LuPos, the configuration of the inputs and outputs is done in code. For a description, see section 9.5.2.

K03 Table index configuration

When GOT commands are used, this parameter determines which inputs of the index are read in for the table positions.

A maximum of 4 inputs (PosMod1, CAN) can be used for this purpose. The entries are made in binary code. the input with the highest number indicates the bit with the highest value.

K04 AUTO / START configuration

With this parameter, the "Automatic" and "Start" inputs can be assigned to the inputs of the of the servocontroller (IS00 and IS01) or the positioning unit (IP10 and IP11) as desired.

K05 Configuration of digital outputs

The local outputs OP10 ... OP13 can either be used as outputs with a fixed function or be used as free outputs by the positioning program. If the output is to fulfil a fixed function, it should be checked off appropriately. If the function is not checked off, the output cannot be freely responded to in the program. The pre-defined functions are:

OP10: Program end
After automatic mode has been selected and as long as no program is being run, this output is set = 1. During the execution of a program, this output is OP10 = 0.

OP11: Axis in position
This output is = 1 if no positioning command is being carried out and the axis is within the positioning window.

OP12: Home position defined
The output is set = 0 as soon as the home position run has been successfully completed or a transfer of the home position has taken place (see also "K70 reference run type").

OP13: Fault signal
The output is set = 1 as soon as the positioning unit is in an error condition and is set back to = 0 after the error has been remedied.

K06 Feed hold configuration

This parameter determines which function the feed release fulfils (if desired).

Axis motions are only allowed if this output is at H level. If this input is set to L level, the feed motion is stopped (interrupted) with the programmed deceleration. The position control remains on.

K07 Read-in hold configuration

This value determines which input the read-in release is applied to (if desired).

By applying an L level to this input in automatic mode, the execution of the program can be interrupted. A positioning task already in progress is completed first.

K08 Jog input configuration

This determines on which two inputs the function of the jog mode should take effect. The axis can then be moved manually with the jog keys at creep feed or rapid feed.

K09 Hardware limit switch configuration

This parameter determines which two inputs the hardware limit switch is connected to. Hardware limit switches should be wired low-active so that they also activate if a line is broken.

5.2 Positioning parameters

K10 Resolution distance nominator

Value range: 0 ... + 2.147.483.647
Unit: Increment

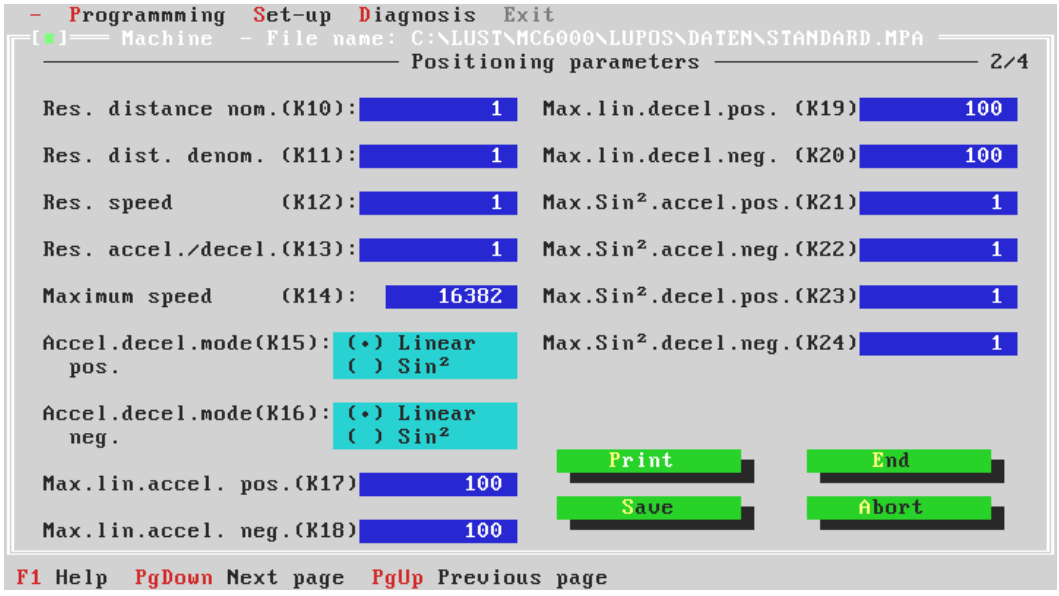
This parameter, together with K11, determines the programming unit for the positioning distance according to the application. $\text{distance unit} = \frac{K10}{K11}$.

Note:
The following applies independently of the encoder used: 1 U_{Motor} ⇔ 65536 increments!

K11 Resolution distance denominator

Value range: 0 ... + 2.147.483.647
Unit: Distance unit

see description of K10



Note:
Because only whole-number values can be entered into the machine parameters, it is inconvenient to program the speed in m/s, for example, because this makes 1 m/s the lowest possible set speed.

K12 Speed resolution

Value range: 0 ... + 2.147.483.647

The speed unit (e.g. m / mm) desired for programming can be set by using this parameter and taking the distance resolution and the gear conversion into account. This means that one can program directly in the unit given here in application programs.

The internal speed unit of >PosMod1 is $\frac{\text{increments}}{5\,\text{ms}}$

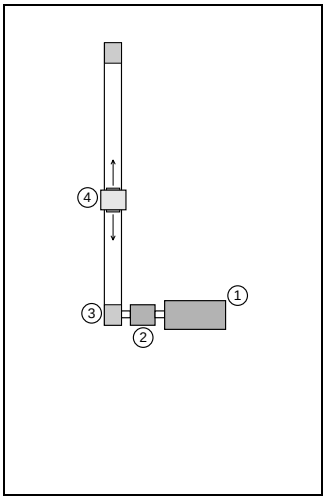
K13 Acceleration resolution

Value range: 0 ... + 2.147.483.647

The unit desired for programming of accelerations can be defined with K13 (e.g. m / s²).

The internal acceleration unit of >PosMod1 is $\frac{\text{increments}}{5\,\text{ms}^2}$

Example of determination of parameters K10 to K13



Linear axis

1. Motor:	1 U_{Motor} ⇔ 65536 increments! 16 bit resolution (applies independently of encoder and motor type)
2. Gears:	i = 4 (1 U_{Axis} ⇔ 4 U_{Motor})
3. Linear conversion:	1 U_{Axis} ⇔ 192 mm
4. slide	

The slide sets the distance during a revolution of the motor.

$$s = \frac{\text{pinion circumference}}{\text{gear translation}} = \frac{2 \cdot \pi \cdot r}{i}$$

$$s = \frac{192\,\text{mm}}{4} = 48\,\text{mm} \text{ back.}$$

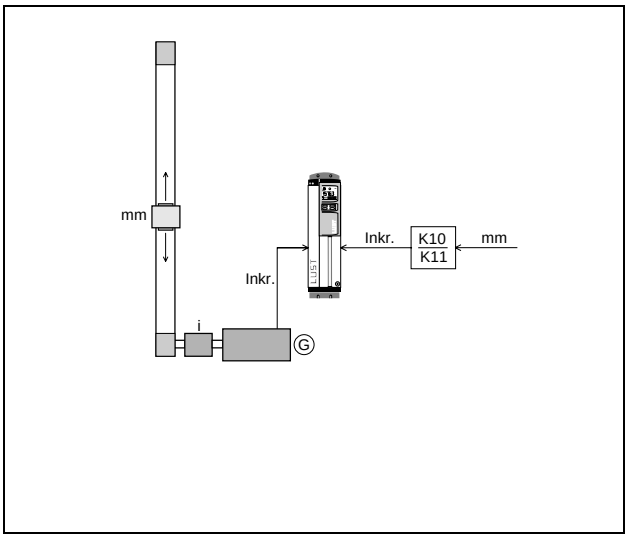
Due to the application, it seems a good idea to program the positioning module in the following units:

- Distance: mm
- Speed: mm/s
- Acceleration: m/s²

The positioning unit works internally with fixed units related to the increments of the encoder.

The desired programming units must be converted into the internal units.

This is done using machine parameters K10 to K13.



Necessary conversions:

	Programming unit		Internal unit
Distance:	e.g. mm	$\xrightarrow{\frac{K10}{K11}}$	$\frac{\text{increments}}{\text{distance unit}}$
Speed:	e.g. $\frac{mm}{s}$	$\xrightarrow{K12}$	$\frac{\text{increments}}{5\,ms}$
Acceleration:	e.g. $\frac{mm}{s^2}$	$\xrightarrow{K13}$	$\frac{\text{increments}}{5\,ms^2}$

Distance resolution (K10, K11):

Generally:

$$\frac{K10}{K11} = x = \frac{65.536 \text{ incr.}}{\text{distance s}}$$

Example: Distance and position indications in mm are desired.

$$\frac{K10}{K11} = x = \frac{65.536 \text{ incr.}}{48\,mm}$$

The proportion K10/K11 should be kept as small as possible. Simplify to obtain:

$$\frac{65.536/16}{48/16} = \frac{4096}{3} \quad \Rightarrow \quad \begin{aligned} K10 &= \underline{\underline{4096}} \\ K11 &= \underline{\underline{3}} \end{aligned}$$

⇒ Position values can only be given in mm. The actual value display is in mm (absolute value in relation to the home position).

Speed resolution (K12):

Generally:

$$\text{programming unit} = K12 \cdot \frac{\text{Incr.}}{5\,\text{ms}}$$

$$K12 = \text{programming unit} \cdot \frac{5\ \text{ms}}{\text{Incr.}}$$

Example: Programming in mm/s is desired

$$K12 = \frac{\text{mm}}{\text{s}} \cdot \frac{5\,\text{ms}}{\text{Incr.}}$$

$$= \frac{5\,\text{ms}}{\text{s}} \cdot \frac{\text{mm}}{\text{Incr.}} \cdot \frac{65.536\ \text{Incr.}}{48\ \text{mm}} \quad \text{with} \quad \frac{5\ \text{ms}}{\text{s}} = \frac{1}{200}$$

$$= \frac{1}{200} \cdot \frac{65.536}{48}$$

$$= 6{,}827 \qquad \Rightarrow \qquad K12 = \underline{\underline{7}}$$

Rounding off K12=7 would give a deviation in the speed indication of $\frac{7}{6{,}827} = 1{,}025 \Rightarrow 2{,}5\%$. (In general, an error of this quuantity can be accepted.)

Solution: Programming the speed in m/s

$$K12 = \frac{\text{m}}{\text{s}} \cdot \frac{5\,\text{ms}}{\text{Incr.}}$$

$$= \frac{5\,\text{ms}}{\text{s}} \cdot \frac{\text{m}}{\text{Inkr.}} \cdot \frac{65.536\ \text{Incr.}}{0{,}048\ \text{m}} \quad \text{mit} \quad \frac{5\ \text{ms}}{\text{s}} = \frac{1}{200}$$

$$= \frac{1}{200} \cdot \frac{65.536}{0{,}048}$$

$$= 6826{,}7 \qquad \Rightarrow \qquad K12 = \underline{\underline{6827}}$$

Error: $\frac{6827}{6826{,}7} \Rightarrow 0{,}005\%$.

but: The speed can now only be given in steps of 1 m/s.

Compromise: Programming the speed in m/min

$$K12 = \frac{\text{m}}{\text{min}} \cdot \frac{5\,\text{ms}}{\text{Incr.}}$$

$$= \frac{5\,\text{ms}}{60\,\text{s}} \cdot \frac{\text{m}}{\text{Incr.}} \cdot \frac{65.536\ \text{Incr.}}{0{,}048\ \text{m}}$$

$$= \frac{0{,}005}{60} \cdot \frac{65.536}{0{,}048}$$

$$= 113{,}8 \qquad \Rightarrow \qquad K12 = \underline{\underline{114}}$$

Error: $\frac{114}{113{,}8} \Rightarrow 0{,}2\%$.

$\Rightarrow$ Indicating the speed in the unit m/min is advantageous, but the other programming units are also possible.

Acceleration resolution (K13):

Generally:

$$\text{programming unit} = K13 \cdot \frac{\text{Incr.}}{(5\,\text{ms})^2}$$

$$K13 = \text{programming unit} \cdot \frac{(5\,\text{ms})^2}{\text{Incr.}}$$

Example: Programming in m/s² is desired

$$K13 = \frac{\text{m}}{\text{s}^2} \cdot \frac{(5\,\text{ms})^2}{\text{Incr.}}$$

$$= \frac{25\,\text{ms}^2}{\text{s}^2} \cdot \frac{\text{m}}{\text{Incr.}} \cdot \frac{65.536\,\text{Incr.}}{0{,}048\,\text{m}} \quad \text{with } 1\,\text{s}^2 = 10^6\,\text{ms}^2$$

$$= \frac{25}{10^6} \cdot \frac{65.536}{0{,}048}$$

$$= 34{,}1 \qquad \Rightarrow \qquad K13 = \underline{\underline{34}}$$

Summary:
Generally, programming the positioning unit in any unit is possible with these machine parameters, for example degrees (for round tables) or throughput over time (conveyor belts).

However, as the example shows, the programming unit must be selected sensibly! Values which are too small must not be selected for the machine parameters or else accuracy is lost, but they cannot be too large either so that no overflows occur.

K14 Maximum speed

Value range: 0 ... Speed limitation of the servomotor MOSNM (_MOT)
Unit: Speed resolution (K12)

This parameter determines the maximum possible positioning speed. If this is exceeded, it is limited to the value entered here.

In the application program, the set speed should not be greater than K14, or else the "Lag distance" error message can arise.

A K14 parameter greater than the speed limitation of the servocontroller cannot be selected. If the limit is exceeded, K14 is set to the maximum authorised value and a corresponding message is displayed.

The servocontroller has a rotational speed control reserve of 10% (max. speed SCSMX = 1,1 x rated motor speed MOSNM), to be able to compensate for positioning errors even at the highest positioning speeds. Asynchronous machines can be operated with POSMOD1 up to the rated speed (not in the field weakening range)!

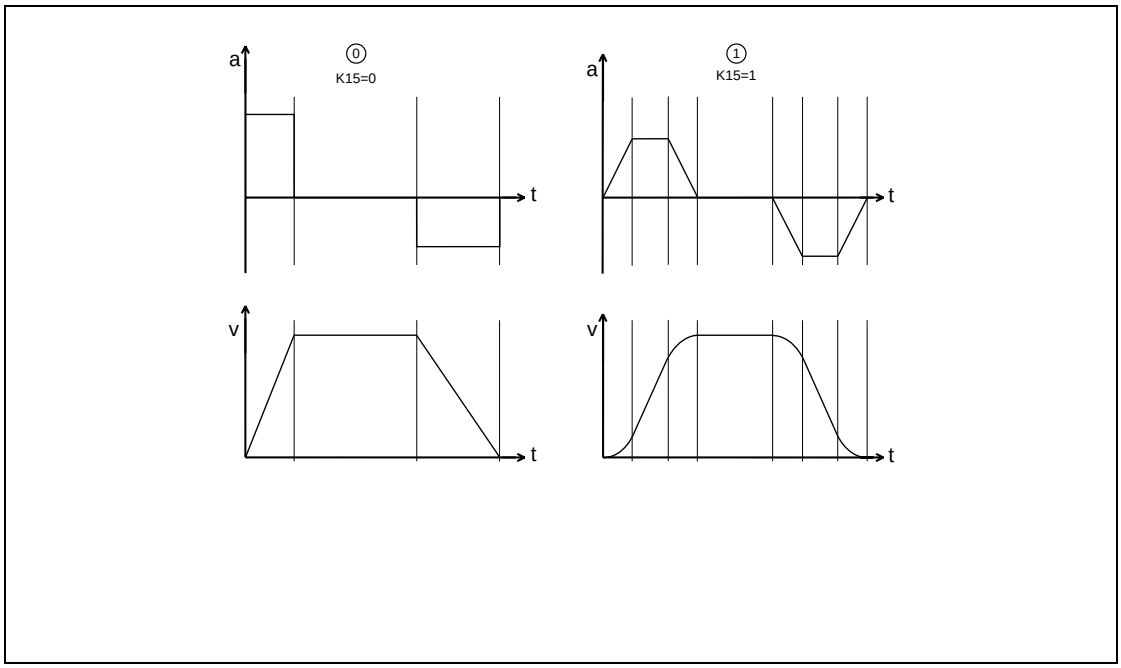
Note:
All accelerations must be selected in such a way that the available torque and the inertia allow this.

K15 Acceleration mode - positive direction

This parameter determines the acceleration type in the positive direction. The linear shape of the acceleration offers the advantage of the greatest possible dynamics. The sin²-shaped acceleration type allows gentle but quick acceleration and braking of the driven axis and therefore positioning with little bucking.

Setting		Acceleration type
0	Linear	Linear acceleration
1	Sin²	Sin²-shaped acceleration

0 - Linear speed course
1 - Sin-shaped speed course (sin² shaped acceleration)



Note:
It is also possible to accelerate sin²-shaped and to brake linearly, for example. To do this, select K15 or K16 = 1 (sin²) and set the initial climb of the braking acceleration (K23 or K24) according to the value of the linear braking deceleration (K19 or K20).

K16 Acceleration mode - negative direction

This parameter determines the acceleration type in the negative positioning direction (see description and table of parameter K15).

K17 Maximum linear approach acceleration - positive direction

Value range: 0 ... + 2.147.483.647
Unit: Acceleration unit (K13)

This parameter determines the maximum authorised approach acceleration (in the positive direction) for the linear acceleration type or in the linear acceleration range for the sin²-shaped acceleration type.

K18 Maximum linear approach acceleration - negative direction

Value range: 0 ... + 2.147.483.647
Unit: Acceleration unit (K13)

This parameter determines the maximum authorised approach acceleration (in the negative direction) for the linear acceleration type or in the linear acceleration range for the sin²-shaped acceleration type.

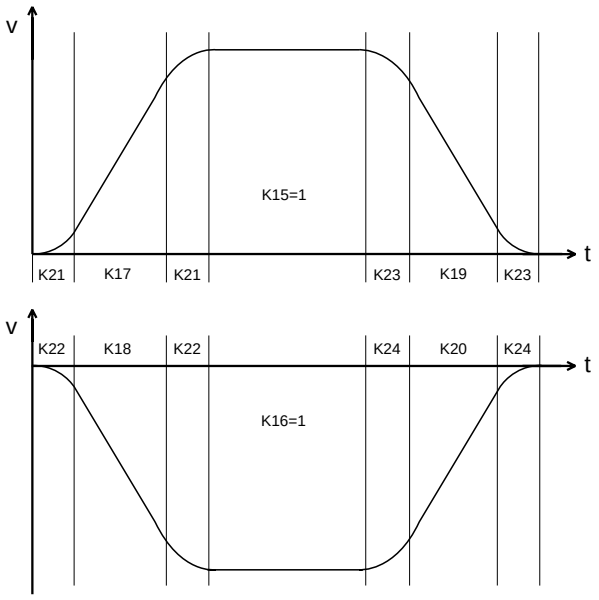
Note on the parameters K17 to K24:

The values defined here are maximum values. In the program, each individual value can be set lower than this maximum value. By using the "SET command", all maximum values can be adjusted by a factor of 1 ... 100 %.

90 ms are required for the recalculation of the acceleration parabolas. Modified values do not take effect until the next positioning task after the axis has halted.

During switching on and after the transfer of the machine parameters through the serial interface, the factors are set to 100%.

K21 and K24 are not significant for the linear acceleration shape.



Calculation of acceleration and braking times for sin²-shaped acceleration (K17 to K24):

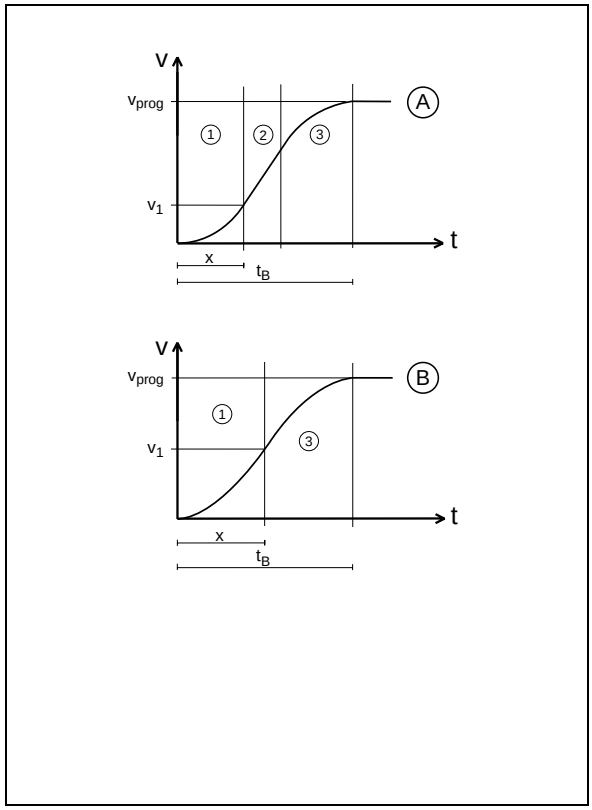
In some applications it is necessary to know or keep to the time until the programmed positioning speed is reached or the time for a positioning task. This is easily done for linear accelerations. The calculations necessary for sin²-shaped acceleration are shown here.

These formulas can also be used to help set the parameters K17 to K24. Additional help is available from the LUPOS user interface: In manual mode, information on the last positioning command can be called with the command %RDS (see Section 9.5.1).

For sin²-shaped approach and braking acceleration, the acceleration phase consists of two mirror image phases (1 and 3) with parabola shaped courses, between which there can be yet another phase (2) with linear acceleration. The maximum duration of phase 1 or 3 is 100 sampling intervals of the reference input, i.e. $100 \cdot 5\text{ms} = 500\text{ms}$.

On the condition that the programmed speed v_{prog} has been reached, a linear phase (Ill. A) arises if either:

- the parameter-set linear acceleration (K17, K18, K19 or K20) was reached during phase 1 (≤ 100 sampling intervals), or
- the speed v_1 after the 100 sampling intervals of phase 1 is less than half of the programmed speed.



Symbols used:

Symbol	Meaning	%RDS
x	Number of phase 1 (3) sampling intervals	yes
v_{prog}	programmed positioning speed	-
v_1	Speed at the end of phase 1	yes
Δv_{lin}	Speed change per 5 ms due the linear acceleration parameter (in speed units)	yes
Δv_{sin}	Speed change per 5 ms at v_1	yes
t_B	Acceleration or braking time	-
s_B	Distance travelled during acceleration	yes

1. Calculation of number of sampling intervals x

$$x = \frac{\Delta v_{sin} + \Delta v_{lin}}{2 \cdot \Delta v_{sin}} \quad \text{with:}$$

Approach pos. direction	$\Delta v_{lin} = K17 \cdot K13 \cdot \{K17 - Faktor\} / 100$
	$\Delta v_{sin} = K21 \cdot K13 \cdot \{K21 - Faktor\} / 100$
Approach neg. direction	$\Delta v_{lin} = K18 \cdot K13 \cdot \{K18 - Faktor\} / 100$
	$\Delta v_{sin} = K22 \cdot K13 \cdot \{K22 - Faktor\} / 100$
Approach neg. direction	$\Delta v_{lin} = K19 \cdot K13 \cdot \{K19 - Faktor\} / 100$
	$\Delta v_{sin} = K23 \cdot K13 \cdot \{K23 - Faktor\} / 100$
Braking neg. direction	$\Delta v_{lin} = K20 \cdot K13 \cdot \{K20 - Faktor\} / 100$
	$\Delta v_{sin} = K24 \cdot K13 \cdot \{K24 - Faktor\} / 100$

2. Calculation of final speed v_1

   a) $x > 100$: $v_1 = 100^2 \cdot \Delta v_{sin}$. Applies for Δv_{lin}: $\Delta v_{lin} = \Delta v_{sin}(x) - \Delta v_{sin(x-1)}$.

   b) $x \leq 100$: $v_1 = x^2 \cdot \Delta v_{sin}$.

3. Calculation of acceleration time t_B

   a) $v_{prog} > 2 \cdot v_1$: with linear phase (2)

$$t_B = \left[(2 \cdot x) + \frac{v_{prog} - (2 \cdot v_1)}{\Delta v_{lin}} \right] \cdot 5ms.$$

   b) $v_{prog} \leq 2 \cdot v_1$: without linear phase

$$t_B = 2 \cdot \sqrt{\frac{v_{prog}}{2 \cdot \Delta v_{lin}}} \cdot 5ms.$$

4. Distance travelled s_B (approaching or braking) is determined with the %RDS command

5. Time for entire positioning procedure t_{tot}

$$s_{tot} = s_{Approad} + s_{v_{prog}} + s_{Broking}$$

$$s_{v_{prog}} = s_{tot} - s_{Approad} - s_{Broking}$$

$$t_{ges} = \frac{s_{v_{prog}}}{v_{prog}}.$$

K19 Maximum linear braking acceleration - positive direction

Value range: 0 ... + 2.147.483.647
Unit: Acceleration unit (K13)

This parameter determines the maximum authorised braking acceleration (in the positive direction) for the linear acceleration type or in the linear acceleration range for the sin²-shaped acceleration type.

K20 Maximum linear braking acceleration - negative direction

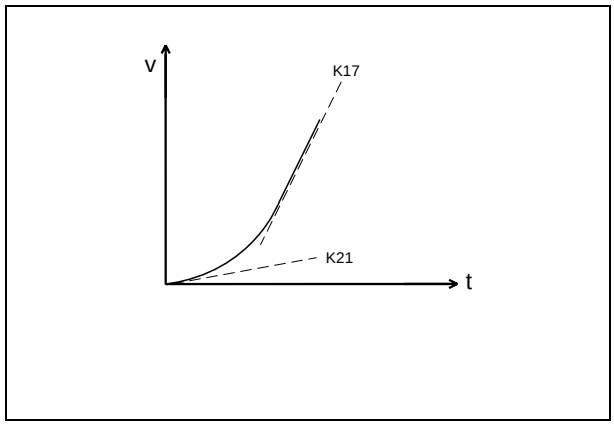
Value range: 0 ... + 2.147.483.647
Unit: Acceleration unit (K13)

This parameter determines the maximum authorised braking acceleration (in the negative direction) for the linear acceleration type or in the linear acceleration range for the sin²-shaped acceleration type.

K21 maximum initial climb for Sin² approach acceleration - positive direction

Value range: 0 ... + 2.147.483.647
Unit: Acceleration unit (K13)

This parameter determines the authorised value of the initial climb of the approach acceleration in the positive direction for sin²-shaped acceleration types.



K22 maximum initial climb for Sin² approach acceleration - negative direction

Value range: 0 ... + 2.147.483.647
Unit: Acceleration unit (K13)

This parameter determines the authorised value of the initial climb of the approach acceleration in the negative direction for sin²-shaped acceleration types.

K23 Maximum initial climb for Sin² braking acceleration - positive direction

Value range: 0 ... + 2.147.483.647
Unit: Acceleration unit (K13)

This parameter determines the authorised value of the initial climb of the braking acceleration in the positive direction for sin²-shaped acceleration types.

K24 Maximum initial climb for Sin² braking acceleration - negative direction

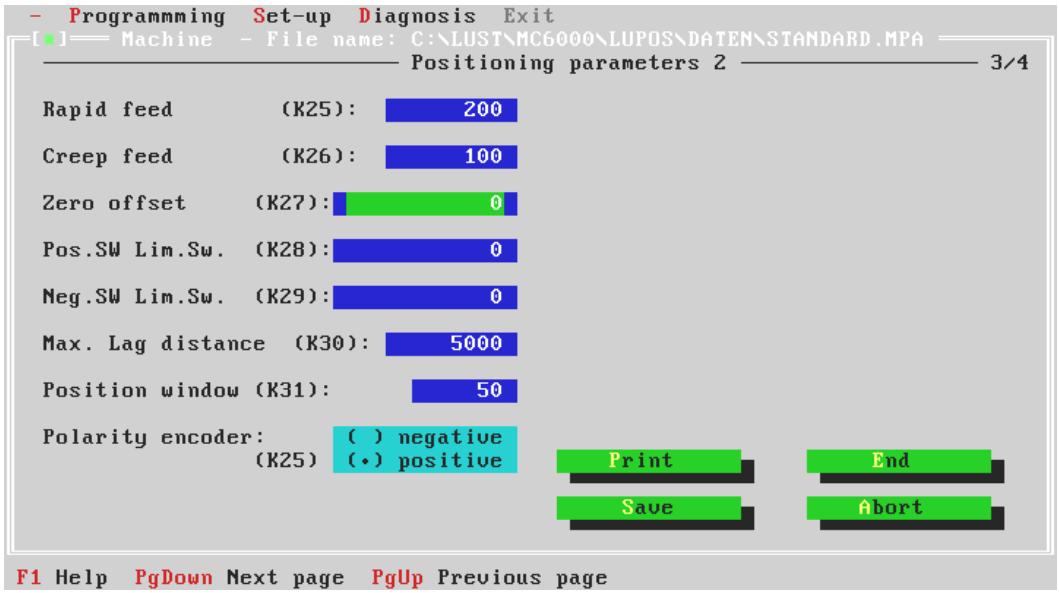
Value range: 0 ... + 2.147.483.647
Unit: Acceleration unit (K13)

This parameter determines the authorised value of the initial climb of the braking acceleration in the negative direction for sin²-shaped acceleration types.

K25 Rapid feed

Value range: 1 ... 9.999.999
Unit: Speed unit (K12)

This parameter determines the speeds applicable in jog mode for rapid feed.



K26 Creep feed

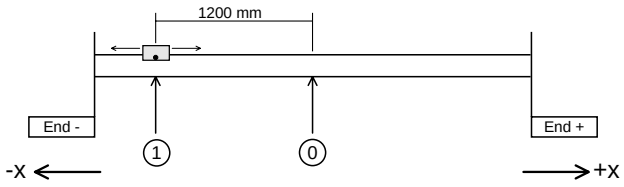
Value range: 1 ... 9.999.999
Unit: Speed unit (K12)

This parameter determines the speeds applicable in jog mode for creep feed.

K27 Zero offset

Value range: - 2.147.483.647 ... + 2.147.483.647
Unit: Distance unit (K10/K11)

This parameter is used to correct the machine zero point in relation to the home position. After a completed reference point run or after transfer of the reference point, the position of the reference point of the current position is set to the value of the zero offset.
Example of the use of zero offset:



Solution: K27 = - 1200 (mm).

0	Machine zero point
1	Home position, current position after completed reference run

The negative sign is due to the fact that the home position is on the left (in the negative axis direction) when viewed from the machine zero point.

K28 Positive software limit switch

Value range: - 2.147.483.647 ... + 2.147.483.647
Unit: Distance unit (K10/K11)

The positioning range limits (in relation to the machine zero point) are defined with the parameters K28 and K29 (except in infinite mode). They are located in front of the positive or negative hardware limit switch, respectively.

If the resulting target position of a positioning command lies outside of these limits, the positioning task is not carried out and an error message is emitted. When a software limit switch is reached, the linear acceleration programmed in the machine parameters is braked and an error message is activated.
If both parameters are set = 0, no monitoring of the software limit switch is carried out (infinite mode).
The software limit switches are not monitored during a reference run.

K29 Negative software limit switch

Value range: - 2.147.483.647 ... + 2.147.483.647
Unit: Distance unit (K10/K11)

For a description, see parameter K28.

K30 Lag distance tolerance

Value range: 0 ... 99.999.999
Unit: Increment

This parameter is the maximum authorised difference between the actual and the set position (±). If this value is exceeded, The "Lag distance" error message (E-FLW) is activated. The reaction to this can be programmed (see MC6000 Servocontroller Operating Instructions, Parameter Description, subject area _SCTY). The factory setting is quickstop.

K31 Position window

Value range: 1 ... 32767
Unit: Increment

This parameter is used to determine whether a positioning command has been completed without a continuation of the program (GOW ...). The actual position is constantly compared with the target position under consideration of the position window.
The output "Axis in position" is set within the position window (if programmed).

K32 Polarity encoder

This parameter determines the sign of the position input. In the "positive" setting, the motor shaft turns to the right in the direction of increasing position values. The "negative" setting allows a reverse of direction, if this seems sensible in the system due to the mechanical coupling.

Setting		Sign
0	negative	Sign change
1	positive	no sign change

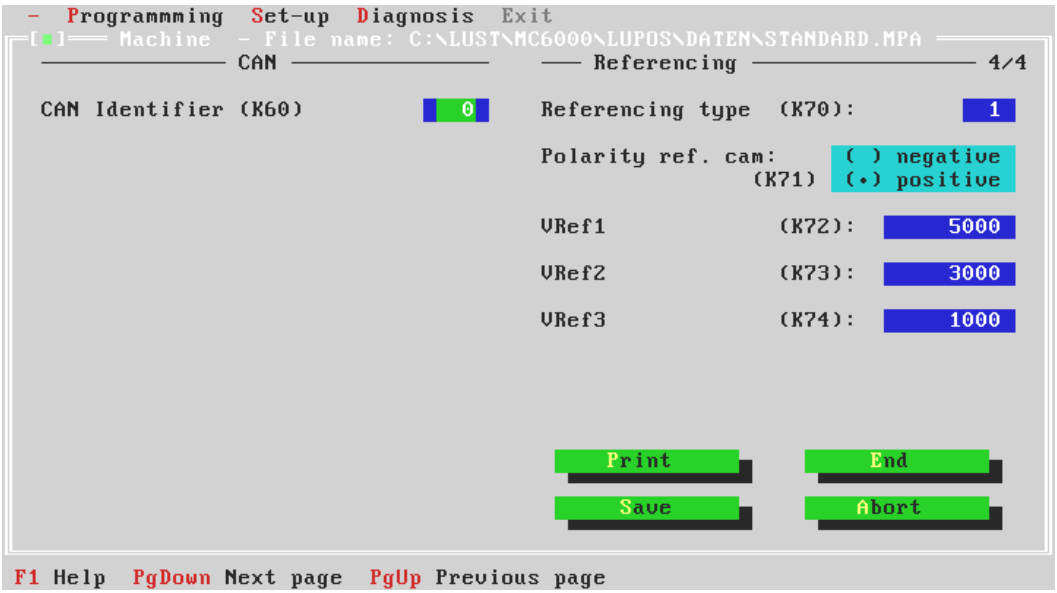
5.3 CAN configuration

K60 CAN I/O- Identifier

Value range: 0 ... 29

The address of the CAN node module is indicated here. It must correspond with the address which was set in the manual mode menu with the command: CAN006=02,99,xx,00,06,00 (xx= Address 00 ... 29) (see chapter...).

The baud rate of the CAN bus is 500 kBit/s.



5.4 Referencing

K70 Reference run type

Depending on the position of the reference cam and the encoder zero impulse, various reference run types can be selected which are described in the following:

The software limit switches are not monitored during a reference run. If a hardware limit switch is approached, the positioning direction is reversed.

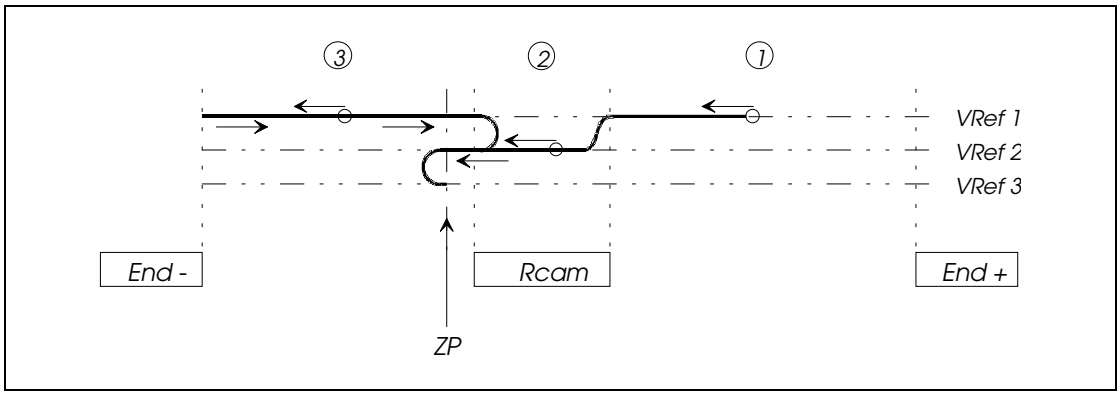
Note: Signal "Home positiondefined"

- For **encoders with zero impulse (G1),** the signal is set when the zero impulse is reached. The positioning profile corresponds to the represented positioning profile.
- For **absolute value encoders (resolvers G2, G3, G5),** the signal is already set when the decisive edge of the reference cam is travelled over. The reference run is then continued at the speed Vref3 up to the zero position of the encoder. When the zero position is reached, the signal "Axis in position" is set as well.

Description of the various reference run types:

Type 0:
No reference run is carried out with this type. Instead, the momentary position is read in and set equal to the zero offset (also under the command SET 0).

Type 1:
The reference cam is located between the two hardware limit switches; the zero impulse to be set is the first one after the cam is left in the negative positioning direction.

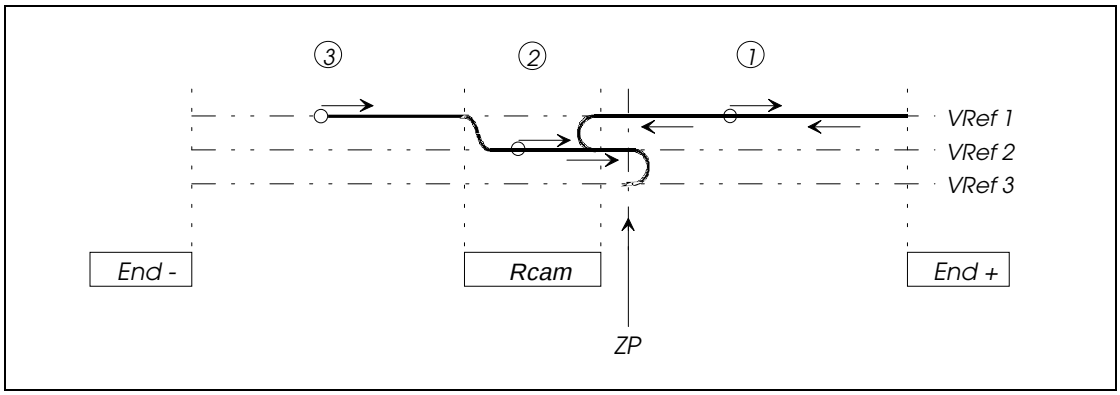


Legend for the graphics:

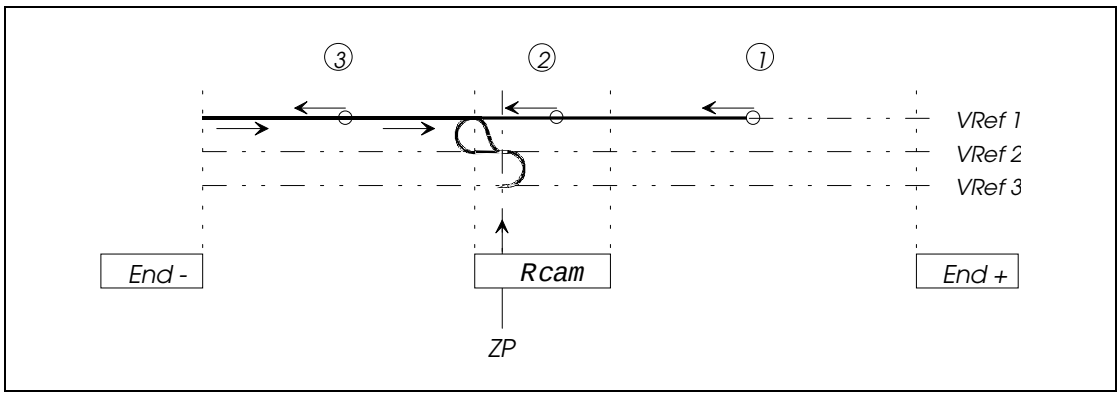
Possible initial positions:		Abbreviations used:		
①	between reference cam and positive limit switch	*End –*	Negative hardware limit switch	(K28)
②	on the reference cam	*End +*	Positive hardware limit switch	(K29)
③	between reference cam and negative limit switch	*RCam*	Reference cam	
		NP	Zero impulse	
		VRef 1	first (highest) Ref., speed	(K72)
		VRef 2	second (middle) Ref. speed	(K73)
		VRef 3	third (lowest) Ref. speed	(K74)

K70 Reference run type (continuation)

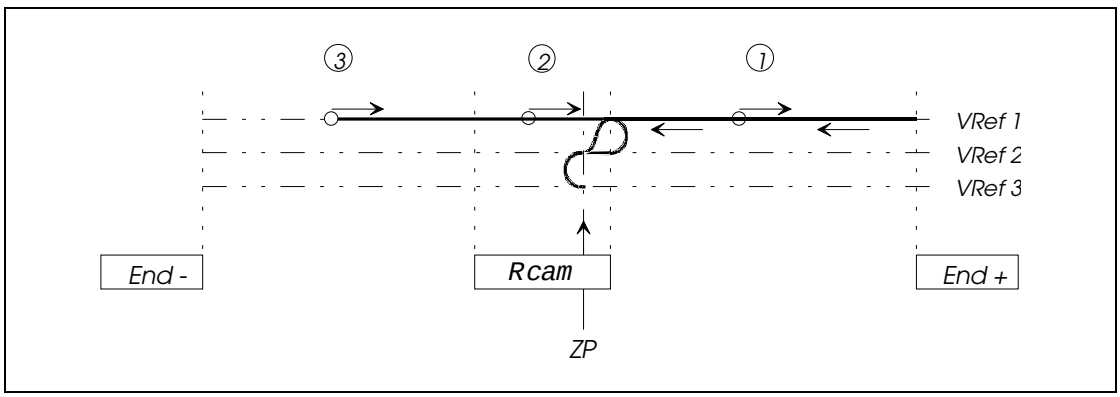
Type 2:
The reference cam is located between the two hardware limit switches; the zero impulse to be set is the first one after the cam is left in the positive positioning direction.



Type 3:
The reference cam is located between the two hardware limit switches; the zero impulse to be set is the first one after the cam is reached in the positive positioning direction.

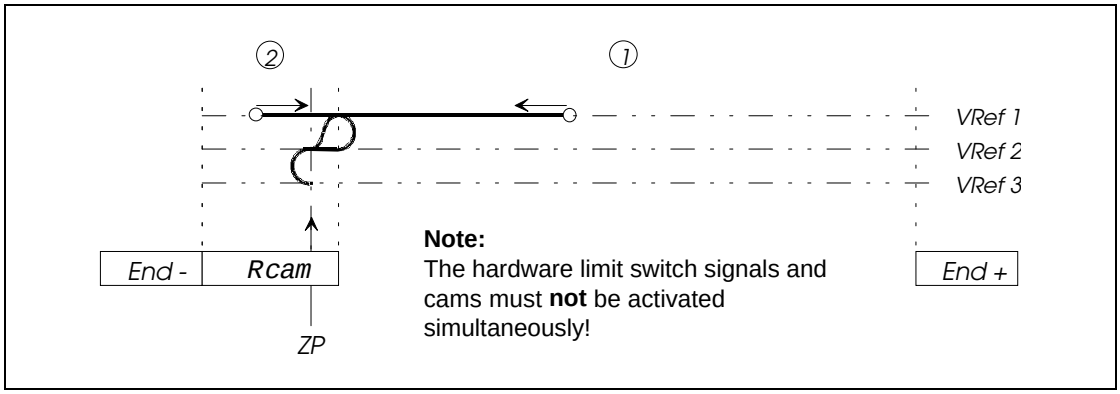


Type 4:
The reference cam is located between the two hardware limit switches; the zero impulse to be set is the first one after the cam is reached in the negative positioning direction.

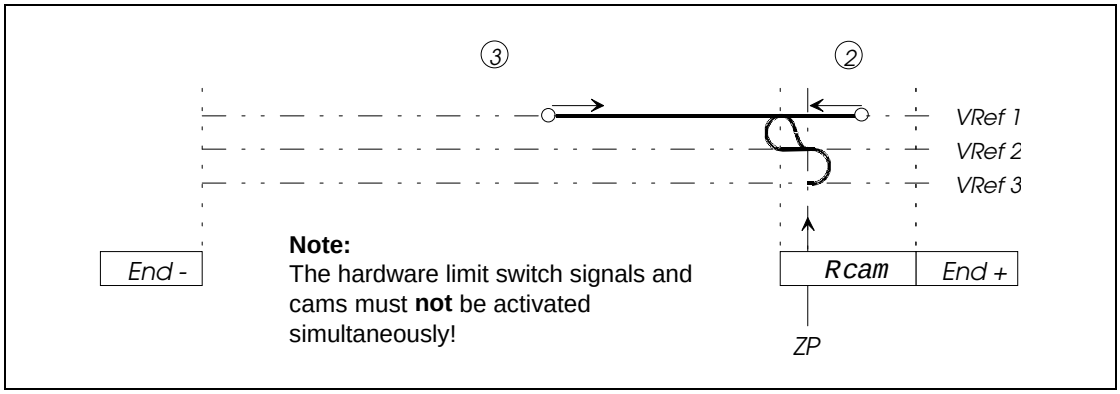


K70 Reference run type (continuation)

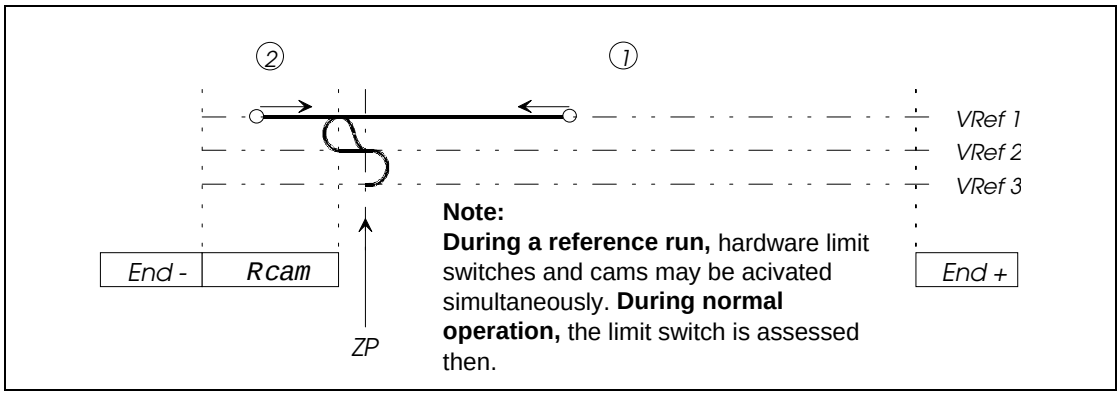
Type 5:
The reference cam is located flush with the negative hardware limit switches; the zero impulse to be set is the first one after the cam is reached in the negative positioning direction.



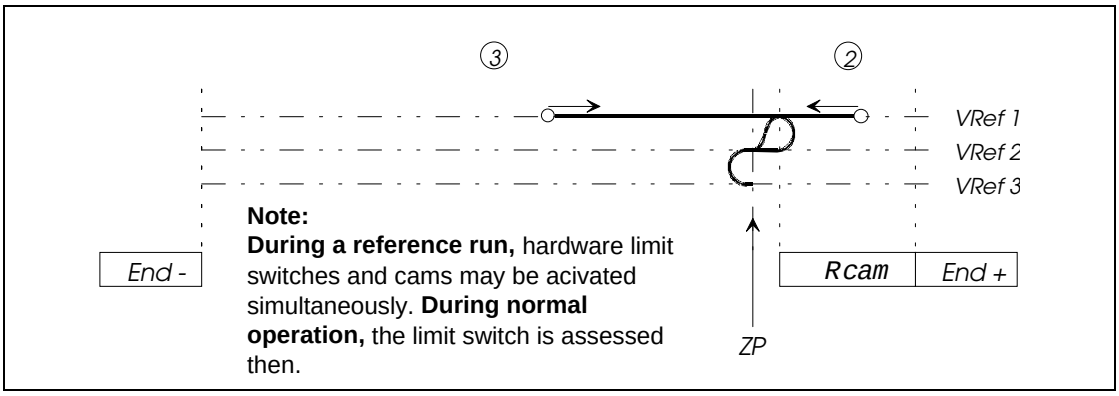
Type 6:
The reference cam is located flush with the positive hardware limit switches; the zero impulse to be set is the first one after the cam is reached in the negative positioning direction.



Type 7:
The reference cam is located flush with the negative hardware limit switches; the zero impulse to be assessed is the first one after the cam is left in the positive positioning direction.



Type 8:
The reference cam is located flush with the positive hardware limit switch; the zero impulse to be assessed should be the first one after the cam is left in the negative positioning direction.



K71 Polarity reference cam

This parameter allows the adaptation of the reference cam with a low or high active output signal.

Setting		Function
0	negative	low-active
1	positive	high-active

K72 VRef 1

Value range: 1 ... 9.999.999
Unit: Speed unit

This parameter determines the first (highest) reference run speed. During programming one should insure that when the reference cam reverses directions, one does not leave on the opposite side (overly large value selected).

K73 VRef 2

Value range: 1 ... 9.999.999
Unit: Speed unit

This parameter determines the second(middle) reference run speed.

K74 VRef 3

Value range: 1 ... 9.999.999
Unit: Speed unit

This parameter determines the third (lowest) reference run speed.

6 Variables, markers and table positions

6.1 Variables

In variables one can store positioning, speed, counter and time values which are used at a later time by the positioning program. Using variables instead of direct input of values is useful above all when a value is used repeatedly. This is often the case for speed values, for example.

POSMOD1 contains a variable memory for 100 variables (H00 ... H99). Variables are whole numbers with an authorised value range of: -2.147.483.647 ... +2.147.483.647.

H99 ... H50	Flash-EPROM[1]
H49 ... H00	RAM[2]

Note:

1	Variables after power outage = value in Flash-EPROM
2	Variables after power outage = 0

After switching on, the variable values from H00 to H49 are set to the value 0. The variables H50 ... H99 are set to the values stored in non-volatile memory (Flash-EPROM)

Note:
To save new variable values in Flash-EPROM, "Store data" in the programming menu must be carried out once.

Within programs, the variables can be influenced with setting commands, e.g. load with a value or calculate with a help magnitude. In manual mode, variables can also be seen and changed from LuPos.

Variables can also be transferred over the serial interface or a bus system at any point in time using the data management commands. Variable values can be transferred completely, partially or individually in this way.

6.2 Markers

Markers can be seen as variables which can only take two conditions: 0 or 1. Markers are used for the further processing of information in user programs.

100 markers are available (M00 ... M99). After switching on, all markers are set = 0. Markers are set in programs with setting commands (SET M ...). Scanning is done with jump commands or subprogram call ups.

Markers maintain their value until they are written over with a new value. In this way, markers which were set in one program can be scanned in another program (e.g. a subprogram).

Markers can also be transferred over the serial interface. This can be done by LuPos or with the "Commands for program and data management" (chapter 9).

6.3 Table positions

The position control can store a table with 16 table positions. Values for positions and positioning distances can be entered in the table positions. The value range for variables is -2.147.483.647 ... +2.147.483.647.

In user programs, one can position to table values using the command GOT.... To do this, it is necessary that the "Table index" contacts appropriately configured inputs of the position control. The table index determines which line of the table (0 ... 15) is used for positioning. The position values can be entered in the programming menu under "Table positions", or can be set in the program (SET command). The 16 positions can be set over a maximum of 4 inputs.

Example:
Four positions should be absolutely approached dependant on two inputs.

IP13 (Bit 1)	IP12 (Bit 0)	Table index	Position
0	0	0	0
0	1	1	100
1	0	2	200
1	1	3	500

Possible programming:

```
N100  GOTA H00:            Approach position at speed from H00
N110  JMP (PW=1) N100:     Wait until position has been reached, then
N120  JMP N110:            ready for new positioning
```

After switching on, all table positions are set to the values stored in flash-EPROM (storing data is necessary!)

Table positions can also be transferred over the serial interface or a bus system like variables and markers can.

If more than 16 table positions are needed, other tables can be loaded using the commands for program and data management, and one can switch between the tables. Or the surplus values are stored in variables and loaded into tables as necessary.

7 Instruction set

The commands described in this chapter can be used in positioning programs for positioning of the driven axes. The positioning programs are created and processed with the editor of the convenient LuPos user interface.

Programming is goal oriented and similar to the widely distributed programming language BASIC. This reduces the time necessary for learning all the commands. This also has the advantage that the positioning program is readable by users without exact knowledge of the PosMod1 instruction set.

The instruction set is divided into the following categories:

- Jump commands
- Subprogram call-ups
- Setting commands
- Positioning and positioning commands
- Wait commands

In addition to the BASIC-type commands, PosMod1 also contains commands which are specially designed for positioning applications. These are not only commands from the category "Positioning and positioning commands", but also, for example, special jump commands whose execution is dependant on the position of the axis.

Jump commands and subprogram call ups can only be used within programs. All other commands can also be called up and executed directly from the user interface for setup (in the "Setup" menu).

Commands are interpreted at call-up. A pre-interpretation is not sensible because, for example, the scanning of the actual position at an earlier point in time would lead to a false result.

The execution time for commands is generally 1 ... 3 ms. Exceptions are pointed out.

The following symbols are used in the command syntax:

Symbol	Meaning	Value range	
Cxx	Counter index	00 ... 99	
Hxx, Hyy	Variable index	00 ... 99	
Kxx	Machine parameters	00 ... 74	
Mxx, Myy	Marker index	00 ... 99	
Ny	Block number	000 ... 999	
Pmm	Mask	00 ... FF Hex	(8 bit)
Pyy	Program number	00 ... 99	
Txx	Table positions	00 ... 15	
Zxx	Timer index	00 ... 07	
b	Value (auxiliary quantity for loading/calculation)	1 ... 65535	(16 bit)
d	Status of a timer or counter	0 ... 65535	(16 bit)
z	Numerical quantity (variables or table positions)	-2.147.483.647 ... +2.147.483.647	(-2,1*10^9 ... +2,1*10^9)

Symbol	Meaning	Value range	
x	Position value [distance unit (K10/K11)]	-2.147.483.647	
y	Speed value [speed unit (K12)]	...	
Ax	Absolute position [distance unit (K10/K11)]		
Rx	Positioning distance [distance unit (K10/K11)]		
Vy	Speed [speed unit]	+2.147.483.647	
END	End of program		
IP	Actual position		
OV	Override (speed norming)		
PW	Position window (with reference to actual value)		
SP	Set position		
ST	Axis stationary (set position = target position)		
UP	Subprogram		
Ippi	Input	PosMod1 = P10 ... P17	
		CAN-Module = C00 ... C43	
Ippi	Input port	PosMod1 = P1	
		CAN-Module = C0 ... C4	
Oppi	Outputs	MC6000 = S00 .. S01	
		PosMod1 = P10 ... P13	
		CAN-Module = C00 ... C27	
Opp	Output port	PosMod1 = P1	
		CAN-Module = C0 ... C2	

7.1 Jump commands and subprogram call-ups

The following is applicable to these commands:
- Use only possible in programs
- Differentiation into conditional and unconditional commands
- Condition for execution indicated in parentheses (....).

Unconditional jump commands/subprogram call-ups are executed in every case (without pre-condition)
Conditional jump commands/subprogram call-ups are only carried out if the indicated condition is fulfilled. The condition for the execution of the command is specified in parentheses (...).

A block number, the program end or a program number is always given as jump target.

Jump to sentence with the number y= 0 .. 999:	JMP (...) Ny
Jump to program end:	JMP (...) END
Call-up of the SP with the number yy= 00 ... 99:	JMP (...) Pyy

Subprograms

A subprogram (SP) is handled as an independent program by the program management. Every SP receives a program number in the range from 00 ... 99. The number of programs and subprograms can therefore total 100. This has the advantage that every program can also be called up as a subprogram.

After execution of the subprogram, the program is continued with the block which follows the call up. The maximum nesting depth for subprograms is 10. If this number is exceeded, an error message is emitted and the program in progress is interrupted.

Unconditional jump commands/subprogram call-ups

These commands are not attached to any pre-conditions (axis position, status of program-internal quantities) and are therefore carried out immediately and unconditionally.

JMP Ny	Jump to block with the number y
JMP END	Jump to program end
JMP Pyy	Call-up of the SP with the number yy

Conditional jump commands/subprogram call-ups

conditional jump commands/subprogram call-ups are attached to a particular condition, which is indicated in parentheses. If the condition is fulfilled, the jump to the specified block number, the program end or the subprogram is carried out. If the condition is not fulfilled, the program is continued with the next sentence.

The conditions described in the following can be attached to the execution of a conditional jump.

Command execution dependant on:

- **Actual position** **Execution time: 5 ... 10 ms**

  Exceed:
  directly: JMP (IP>x) Ny/END/Pyy
  with variable: JMP (IP>Hxx) Ny/END/Pyy

  Fall below:
  directly: JMP (IP>x) Ny/END/Pyy
  with variable: JMP (IP<Hxx) Ny/END/Pyy

  x = Comparative position [distance unit]
  Hxx = Variable index (00 ... 99)
  Ny = Block number (000 ... 999)

- **Set position** **Execution time: 5 ... 10 ms**

  reached:
  directly: JMP (IP>x) Ny/END/Pyy
  with variable: JMP (IP>Hxx) Ny/END/Pyy

  Exceed:
  directly: JMP (IP>x) Ny/END/Pyy
  with variable: JMP (IP>Hxx) Ny/END/Pyy

  Fall below:
  directly: JMP (SP<x) Ny/END/Pyy
  with variable: JMP (SP<Hxx) Ny/END/Pyy

  x = Comparative position [distance unit]
  Hxx = Variable index (00 ... 99)
  Ny = Block number (000 ... 999)

- **Axis status**

  PW reached: JMP (PW = 1) Ny/END/Pyy Actual position in position window [1)]
  PW not reached: JMP (PW = 0) Ny/END/Pyy IP not position window

  Axis stationary: JMP (ST = 1) Ny/END/Pyy Set position = target position [2)]
  Axis moving: JMP (ST = 0) Ny/END/Pyy Set position ≠ target position

  Ny = Block number (000 ... 999)

  [1)] Positioning completed, "Axis in position" output is set,
  Execution time: 5 ... 10 ms

  [2)] Positioning arithmetically complete

Command execution dependant on: (continuation)

- **Status of digital input**

  Status = 0: JMP (Ippi = 0) Ny/END/Pyy
  Status = 1: JMP (Ippi = 1) Ny/END/Pyy

Ippi	=	Local inputs	(IP10 ... IP17)
	=	External inputs	(IC00 ... IC77)
Ny	=	Block number	(000 ... 999)

- **Status of a logical marker**

  directly: JMP (Mxx= 0) Ny/END/Pyy
  JMP (Mxx= 1) Ny/END/Pyy

  Indexed: JMP (M[Cxx] = 0) Ny/END/Pyy
  JMP (M[Cxx] = 1) Ny/END/Pyy

Cxx	=	Counter index	(00 ... 99)
Mxx	=	Marker index	(00 ... 99)
Ny	=	Block number	(000 ... 999)

- **Status of a counter**

  Compare: JMP (Cxx = d) Ny/END/Pyy
  Exceed: JMP (Cxx > d) Ny/END/Pyy
  Fall below: JMP (Cxx < d) Ny/END/Pyy

d	=	Comparative counter status	(0...65535)
Cxx	=	Counter index	(00 ... 99)
Ny	=	Block number	(000 ... 999)

- **Quantity of a variable (direct comparison)**

  Compare:
  direct: JMP (Hxx = z) Ny/END/Pyy
  Indexed: JMP (H[Cxx] = z) Ny/END/Pyy

  Exceed:
  direct: JMP (Hxx > z) Ny/END/Pyy
  Indexed: JMP (H[Cxx] > z) Ny/END/Pyy

  Fall below:
  direct: JMP (Hxx < z) Ny/END/Pyy
  Indexed: JMP (H[Cxx] < z) Ny/END/Pyy

z	=	Variable value	($-2{,}1 * 10^9$ ... $+2{,}1 * 10^9$)
Cxx	=	Counter index	(00 ... 99)
Hxx	=	Variable index	(00 ... 99)
Ny	=	Block number	(000 ... 999)

Command execution dependant on: (continuation)

- **Quantity of a variable (comparison with second variable)**

  Compare:

  direct: JMP (Hxx = Hyy) Ny/END/Pyy
  Indexed: JMP (H[Cxx] = Hyy) Ny/END/Pyy

  Exceed:

  direct: JMP (Hxx > Hyy) Ny/END/Pyy
  Indexed: JMP (H[Cxx] > Hyy) Ny/END/Pyy

  Fall below:

  direct: JMP (Hxx < Hyy) Ny/END/Pyy
  Indexed: JMP (H[Cxx] < Hyy) Ny/END/Pyy

Cxx	=	Counter index	(00 ... 99)
Hxx, Hyy	=	Variable index	(00 ... 99)
Ny	=	Block number	(000 ... 999)

- **Status of a timer (time counter)**

  Compare: JMP (Zxx = 0) Ny/END/Pyy Timer run out? *)
  Exceed: JMP (Zxx > d) Ny/END/Pyy
  Fall below: JMP (Zxx < d) Ny/END/Pyy

d	=	Comparative timer status	(0...65535)
Ny	=	Block number	(000 ... 999)
Zxx	=	Timer index	(00 ... 07)

  *) Note:
  An inquiry at equal values is only possible when the timer has run out (i.e. "=0") because it is not guarantied that a particular intermediate status ("=d") has been reached at the moment of inquiry.

7.2 Setting commands

Various operations in the positioning programs can be carried out using the setting commands.

- Setting of outputs (directly, with masks, with logical links ...)
- Setting of markers (directly, indexed, with logical links, ...)
- Loading values, calculating, ...
- Loading counters, incrementing, decrementing
- Setting timers and starting
- Setting table positions
- Switching override on and off
- Changing machine parameters (acceleration parameters)

Setting commands can also be entered and directly executed in the "Setup" menu of the user interface.

- **Setting digital output/output port**

Direct:	SET Oppi = 0		
	SET Oppi = 1		
With markers:	SET Oppi = Myy		
With masks:			
Direct:	SET Opp = Pmm		
With counters:	SET Opp = Cxx		
With logical link:			
OR:	SET Oppi = (Ippi | Iqqk)	Link of two inputs	
	SET Oppi = (Ippi | Myy)	Link of input and marker	
	SET Oppi = (Mxx | Myy)	Link of two markers	
AND:	SET Oppi = (Ippi & Iqqk)	Link of two inputs	
	SET Oppi = (Ippi & Myy)	Link of input and marker	
	SET Oppi = (Mxx & Myy)	Link of two markers	

Mxx, Myy	=	Marker index	(00 ... 99)
Pmm	=	Mask to be given out	(00 ... FF Hex)
Cxx	=	Counter index	(00 ... 99)
Ippi, Iqqk	=	Inputs	(P10 ... P17 = PosMod1)
			(C00 ... C43 = CAN modules)
Oppi	=	Outputs	(S00 ... S01 = MC6000)
			(P10 ... P13 = PosMod1)
			(C00 ... C27 = CAN modules)
Opp	=	Output port	(P1 = local)
			(C0 ... C2 = CAN modules)

The symbol "I" is ASCII symbol no. 124. It is obtained by holding down the <ALT> key and entering "124" on the numerical keypad at the same time.

- **Setting logical markers**

Direct: SET Mxx = 0
SET Mxx = 1

Indexed: SET M[Cxx] = 0
SET M[Cxx] = 1

With two markers:

Direct:	SET Mxx = Myy	
	SET Mxx = -Myy	Invert markers
Indexed:	SET M[Cxx] = Myy	
	SET M[Cxx] = -Myy	Invert markers

With logical link:

OR:

Direct:	SET Mxx = (Ippi | Iqqk)	Link of two inputs
	SET Mxx = (Ippi | Myy)	Link of input and marker
Indexed:	SET M[Cxx] = (Ippi | Iqqk)	Link of two inputs
	SET M[Cxx] = (Ippi | Myy)	Link of input and marker

AND:

Direct:	SET Mxx = (Ippi & Iqqk)	Link of two inputs
	SET Mxx = (Ippi & Myy)	Link of input and marker
Indexed:	SET M[Cxx] = (Ippi & Iqqk)	Link of two inputs
	SET M[Cxx] = (Ippi & Myy)	Link of input and marker

Cxx	=	Counter index	(00 ... 99)
Mxx, Myy	=	Marker index	(00 ... 99)
Ippi, Iqqk	=	Inputs	(P10 ... P17 = local) (C00 ... C77 = CAN modules)

The symbol "|" is ASCII symbol no. 124. It is obtained by holding down the <ALT> key and entering "124" on the numerical keypad at the same time.

- **Setting variables**

Direct: SET Hxx = z

Indexed: SET H[Cxx] = z

With 2 variables:

Direct:	SET Hxx = Hyy
Indexed:	SET H[Cxx] = Hyy

With two indexed variables:

Direct:	SET Hxx = H[Cyy]
Indexed:	SET H[Cxx] = H[Cyy]

At counter reading:

Direct:	SET Hxx = Cyy
Indexed:	SET H[Cxx] = Cyy

- **Setting variables (continuation)**

By calculation - directly: [2])

Addition	SET Hxx + z	
Subtraction	SET Hxx - z	
Multiplication	SET Hxx * z	
Division	SET Hxx : z	z ≠ 0 [1])

By calculation -indexed: [2])

Addition	SET H[Cxx] + z	
Subtraction	SET H[Cxx] - z	
Multiplication	SET H[Cxx] * z	
Division	SET H[Cxx] : z	z ≠ 0 [1])

Calculation with second variable - directly: [2])

Addition	SET Hxx + Hyy	
Subtraction	SET Hxx - Hyy	
Multiplication	SET Hxx * Hyy	
Division	SET Hxx : Hyy	Hyy ≠ 0 [1])

Calculation with second variable - directly: [2])

Addition	SET H[Cxx] + Hyy	
Subtraction	SET H[Cxx] - Hyy	
Multiplication	SET H[Cxx] * Hyy	
Division	SET H[Cxx] : Hyy	Hyy ≠ 0 [1])

With actual position:

Direct:	SET Hxx = IP
Indexed:	SET H[Cxx] = IP

With set position:

Direct:	SET Hxx = SP
Indexed:	SET H[Cxx] = SP

With override value:

Direct:	SET Hxx = OV
Indexed:	SET H[Cxx] = OV

With table position:

Direct:	SET Hxx = Txx
Indexed:	SET H[Cxx] = Txx

z	=	Variable value	(-2,1 *10^9 ... +2,1 * 10^9)
Cxx, Cyy	=	Counter index	(00 ... 99)
Hxx, Hyy	=	Variable index	(00 ... 99)
Txx	=	Table index	(00 ... 15)

[1]) z or Hyy = 0 is not allowed (division by 0!)
(error message is activated)

[2]) For these operations, insure that no range overflow occurs.

- **Setting counters**

  Direct: SET Cxx = d

  With variable: SET Cxx = Hyy

  With counter: SET Cxx = Cyy

  Incrementing/decrementing counters:
  SET Cxx + d
  SET Cxx - d

  Incrementing/decrementing counters with variables:
  SET Cxx + Hyy
  SET Cxx - Hyy

  With input port: SET Cxx = Ipp

d	=	Counter value	(0 ... 65535)
b	=	Add./subtr. auxiliary quantity	(1 ... 65535)
Cxx, Cyy	=	Counter index	(00 ... 99)
Hyy	=	Variable index	(00 ... 99)
Ippi	=	Input port	(P1 = PosMod1) (C0 ... C4 = CAN-Module)

- **Setting table position**

  Direct: SET Txx = z
  Indexed: SET T[Cxx] = z

  Direct:
  With variable: SET Txx = Hyy
  With actual position: SET Txx = IP
  With set position: SET Txx = SP

  Indexed:
  With variable: SET T[Cxx] = Hyy
  With actual position: SET T[Cxx] = IP
  With set position: SET T[Cxx] = SP

z	= Value of the table position	(-2,1 *10^9 ... +2,1 * 10^9)
Cxx	= counter index	(00 ... 99)
Hyy	= variable index	(00 ... 99)
Txx	= index of the table positions	(00..15)

- **Transferring home position**

  The execution of this command sets the current actual position to the zero offset value defined in the machine parameters.

  SET O

- **Setting and starting timers (time counters)**
  After the time is loaded with a value, the value is automatically decreased by one every millisecond until finally the value zero is reached.

  Direct: SET Zxx = d
  With variable: SET Zxx = Hyy

  d = timer value (0 ... 65535)
  Hyy = variable index (00 ... 99)
  Zxx = timer index (00 ... 07)

- **Switching override on and off**
  When the override is switched on, the programmed speed for all positioning tasks is multiplied by the override (0 ... 150%) transferred from the servocontroller.
  When the override is switched off, positioning is always done at the programmed speed (corresponds to override = 100%).

  Switch on: SET OV = 1
  Switch off: SET OV = 0

  Note:

Acceleration type:	**Transfer of a modified override value:**
Linear	Immediately during positioning
Sin²-shaped	At next positioning command

- **Changing acceleration type Execution time: up to 90 ms**
  The acceleration type can be changed while the unit is standing still with the following setting commands:

  Acceleration type for the positive direction:
  SET K15 = 0/1 (0 = linear; 1 = sin²)
  With variable: SET K15 = Hxx (Hxx = 0/1)

  Acceleration type for the negative direction:
  SET K16 = 0/1 (0 = linear; 1 = sin²)
  With variable: SET K16 = Hxx (Hxx = 0/1)

- **Changing the maximum acceleration values**
  The values indicated in the machine parameters for acceleration K17 ... K24 correspond to 100%. By using appropriate SET commands, these values can be set to a value between 1 and 100% of the machine parameter. The changes do not take effect until the next positioning task after the axis has halted.
  Execution time: up to 90 ms (program not resumed until after calculation).

  Direct: SET Kxx = p
  With variable: SET Kxx = Hxx

  p = percentage value of the acceleration (0 ... 100%)
  Kxx = number of the machine parameter (17 ... 24)
  Hxx = variable index (00 ... 99)

  Example: Setting linear approach acceleration in the positive direction to 50%
  SET K17 = 50

7.3 Positioning commands

The driven positioning axis can be moved with these commands. They can also be used in manual mode (setup menu) to move the axis in individual step mode.

A distinction can be made between two methods of moving the axis:
1. Moving to a particular position (absolute positioning)
2. Moving over a particular distance (relative positioning)

In the commands, absolute positioning is designated with "A" , and relative positioning is designated with "R".

The absolute position or the positioning distance and the speed is entered either as a fixed value or as variables in any combination. Table positions can also be used to enter the position or the distance. In this case, the axis moves at the transferred speed to the value of the table position whose index can be read in at the configured input terminals.

Positioning commands can be executed either with or without program continuation.

- With program continuation (GO ...)
  If such a command is given within a program, the program is immediately continued with the next block after the axis is started. In this way, several commands can be processed in parallel.
  If the command is given while a positioning task is in progress, positioning to the new target position is done at the modified speed. The new command is executed immediately, i.e. the position from the original command is no longer approached!

- Without program continuation (GOW ...)
  With these commands, the subsequent block is not executed until the actual position has reached the positioning window. The program will not be continued as long as the axis is not in the positioning window (e.g. due to a contouring error).

  "W" stands for "wait". GOW = "go and wait".

- **Positioning with continuation**

  Position or distance directly/speed directly
  Absolute: GO Ax Vy
  Relative: GO Rx Vy

  Position or distance directly/speed with variable
  Absolute: GO Ax Hyy
  Relative: GO Rx Hyy

  Position or distance with variable/speed directly
  Absolute: GO A Hxx Vy
  Relative: GO R Hxx Vy

  Position or distance with variable/speed with variable
  Absolute: GO A Hxx Hyy
  Relative: GO R Hxx Hyy

Position or distance with table/speed directly
Absolute: GOTA Vy
Relative: GOTR Vy

Position or distance with table/speed with variable
Absolute: GOTA Hyy
Relative: GOTR Hyy

Ax = Absolute position [distance unit]
Rx = Positioning distance [distance unit]
Hxx = Index of variables with position value
Hyy = Index of variables with speed value
Vy = Speed value [speed unit]

- **Positioning without continuation**

Position or distance directly/speed directly
Absolute: GOW Ax Vy
Relative: GOW Rx Vy

Position or distance directly/speed with variable
Absolute: GOW Ax Hyy
Relative: GOW Rx Hyy

Position or distance with variable/speed directly
Absolute: GOW A Hxx Vy
Relative: GOW R Hxx Vy

Position or distance with variable/speed with variable
Absolute: GOW A Hxx Hyy
Relative: GOW R Hxx Hyy

Position or distance with table/speed directly
Absolute: GOTWA Vy
Relative: GOTWR Vy

Position or distance with table/speed with variable
Absolute: GOTWA Hyy
Relative: GOTWR Hyy

Ax = Absolute position [distance unit]
Rx = Positioning distance [distance unit]
Hxx = Index of variables with position value
Hyy = Index of variables with speed value
Vy = Speed value [speed unit]

- **Reference running:**
  The reference run is carried out using the reference run type defined in the machine parameters (K70) and the corresponding speeds (K72 ... .

  If this command is given within a program, the subsequent block does not take effect until after the reference run has ended.

  GO 0

- **Infinite moving**

  Direct: GO V+y — Positive direction
  GO V-y — Negative direction

  with variable: GO V Hxx

  Hxx = Index of variables with speed value
  The sign of the value in Hxx determines the positioning direction.
  y = Speed value [speed unit]

- **Braking**
  For normal braking at the programmed acceleration (in accordance with machine parameters K15 to K24):

  STOP B

  For quick braking (e.g. emergency braking) at maximum deceleration (linear in accordance with K19 or K20, even if $\sin^2$-shaped acceleration is selected):

  STOP M

- **Braking and switching off the position control system**
  Quick braking (speed set point = 0) and subsequent position control switch-off (e.g. to set parameters for the servocontroller):

  STOP 0

  Switching off the control system switches off the current to the motor. This is not allowed in all applications without further measures (e.g. in lifting applications).
  The user must insure that the unit is not damaged and that persons are not injured!

  Switch the position control back on with:
  STOP B or
  STOP M

7.4 Wait commands and storing

- **Time**
  With these commands, one can implement a delay of a particular time in milliseconds. After this time has expired, the program is continued with the next sentence.

  Direct: WAIT b
  with variable: WAIT Hxx

  b = Waiting time in [ms] (1 ... 65535)
  Hxx = index of variables with waiting time

- **Axis status**
  The program is not continued until the following condition is fulfilled:

  PW reached: WAIT PW Actual position in position window [1)]

  Axis stationary: WAIT ST Set position = target position [2)]

  [1)] Positioning completed, "Axis in position" output is set
  [2)] Positioning arithmetically complete

- **Input status**
  The programs not continued until the input concerned shows the expected status (high or low level).

  WAIT (Ippi = 0)
  WAIT (Ippi = 1)

  Ippi = Input (P10... P17 = PosMod1)
  (C00 ... C43 = CAN modules)

- **Storing**
  Securing data (programs, machine parameters, variables H50 to H99 and table positions) in flash EPROM can also be activated by a program. Securing can only be carried out while the axis is stationary, or else error 35, "Unauthorised command during axis motion", is announced. It takes 6 seconds until the next command is executed.

  SAVE

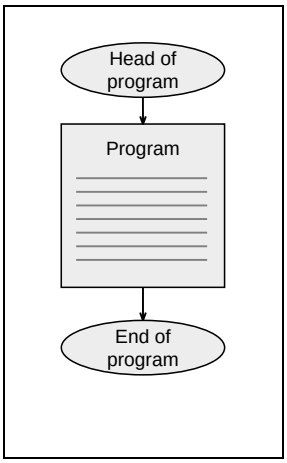
  This command can be sensibly used to secure data even without a PC, e.g. for a teachin (recording table positions).

8 Programming

This chapter describes the creation of user programs using several examples.

8.1 Program layout

A program is divided into a program header, the actual program section and the program end.



A **program header** consists of a line which contains the program number and usually a program name as well. The name can have a maximum of 20 characters.

```
        %Pyy  (name)
e.g.:   %P00  (test program)
```

The lines of the actual **program section** are called program blocks. A program can have a maximum of 1000 program blocks (N0 ... N999). The maximum number of programmable blocks is limited to 3000.
Every programming block consists of a line number, the command and one or two operands.

```
N050  WAIT ST
```

If a comment is to be inserted, it should be separated with a semicolon:

```
N050  WAIT ST     ;wait until axis is stationary
```

This line (without line number) is always at the **program end:**

```
END
```

It is advisable to number the program blocks in steps of 10 (010, 020, ...) to be able to insert lines later.

Place a space between block numbers, commands and operands (no tabs).
See program editor, chapter 4.5, for more information.

From a program (main program=MP) one can also call up further programs (subprograms=SP). The maximum nesting depth for UP is 10. Subprograms are constructed and handled in the same way as main programs.

The advantages of using subprograms are above all due to:

- the orderly program structure and
- The external placement of repeatedly used program sections.

8.2 Programming examples

The numerical values for distance, speed and acceleration always refer to the programming unit stipulated in the machine parameters.

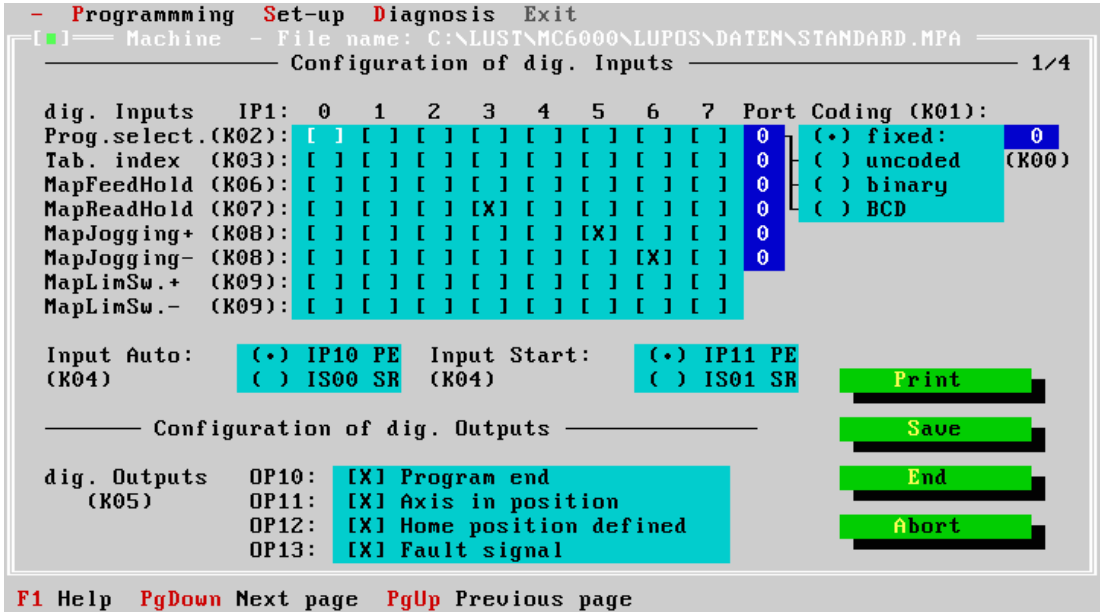
Example 1: Conveyor belt

After starting the drive of a conveyor belt is to move 1 m (corresponds to 10 motor revolutions) at a speed of 20 RPM (of the motor). after the waiting time of 5 s, the procedure is to be repeated until the input is set back.

Program (printout with program editor):

```
Program - File-Name:  C:\LUPOS\DATEN\BSP1.PRG

Friday, January 20th, 1995 8:01

%P00 (Beispiel 1)
N010 SET 0;              Define home position
N020 JMP (IP12=0)N020;   Further if input = high
N030 GOWR10 V20;         Traverse in positive direction at 20 RPM
N040 WAIT 5000;          Wait 5s
N050 JMP N020;           Restart cycle
END
```

Machine parameters (1st screen):



Machine parameters (printout with machine parameter screen):

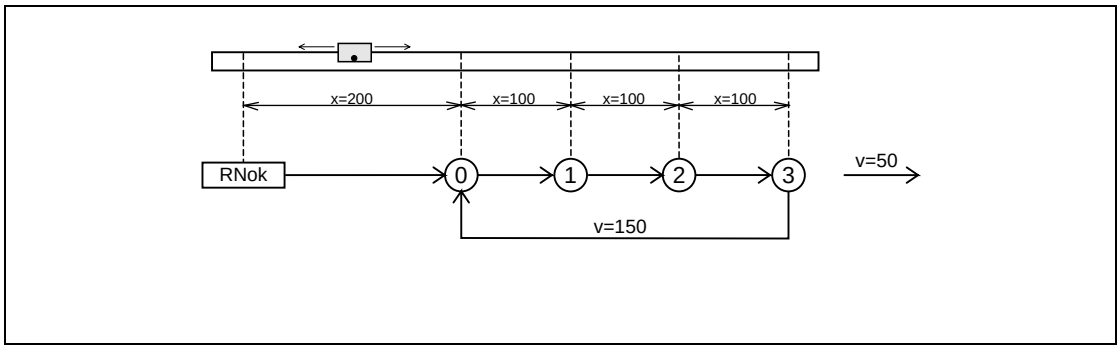
```
Machine param.- File-Name: C:\LUPOS\DATEN\BSP1.MPA

Friday, January 20th, 1995     7:48
```

			Explanation
K00:	Fixed program number:	0	P00 is started
K01:	Program number coding:	0	uncoded
K02:	Program number selection:	0000	not connected [1]
K03:	Table index selection:	0000	not connected [1]
K04:	AUTO/START configuration:	00	IP10 and IP11 (PosMod1)
K05:	Configuration of dig outputs:	00CF	All with defined function [1]
K06:	Map feed hold:	0000	not connected [1]
K07:	Map read-in hold:	0000	not connected [1]
K08:	Map jogging positive/negative:	00200040	IP15 and IP16 [1]
K09:	Map hardware limit switch pos./neg.	00000000	not connected (conveyor belt)[1]
K10:	Distance resolution counter:	65536	Distance unit = 1 revolution
K11:	Distance resolution denominator:	1	Distance unit = 1 revolution
K12:	Speed resolution:	5	Speed unit = 1 RPM
K13:	Acceleration resolution:	1	not connected
K14:	Max. speed:	3000	
K15:	Acceleration type positive:	0	Linear
K16:	Acceleration type negative:	0	Linear
K17:	Max linear initial accel. +	10	
K18:	Max linear initial accel. -	10	
K19:	Max linear braking decel. +	10	
K20	Max. Lin. braking decel. -	10	
K21:	Max sin initial accel. +	5	
K22:	Max sin initial accel. -	5	
K23:	Max sin braking decel. +	5	
K24:	Max. sin braking decel. +	5	
K25:	Speed at rapid feed:	200	
K26:	Speed at creep feed:	100	
K27:	Zero offset:	0	
K28:	Pos. software limit switch:	0	not connected (conveyor belt)
K29:	Neg. software limit switch:	0	not connected (conveyor belt)
K30:	Lag distance tolerance:	5000	
K31:	Position window:	50	
K32:	Direction sign:	1	Positive (no sign change)
K60:	CAN node number:	0	
K70:	Reference run type:	0	Current position = reference position
K71:	Reference cam polarity:	1	Positive (high-active)
K72:	Reference run speed 1:	500	
K73:	Reference run speed 2:	100	
K74:	Reference run speed 3:	50	

[1] The configuration of the inputs and outputs is represented in code in the printout. For description, see section 9.5.2, "Coded Representation for Inputs and Outputs ".

Example 2: Approaching positions absolutely

The four positions should be approached at the speed v=50 and waited at for 1 s. Three times the speed is to be used for the motion returning to the initial position.



Positions and speeds are input directly as values. Acceleration is input according to machine parameters.

```
%P01 (Beispiel 1)
N010 GO 0;           Reference run [1)]
N030 GO A200 V50;    Approach initial position
N040 WAIT ST;        Wait until axis is stationary
N050 WAIT 1000;      Wait 1 s
N060 GOW A300 V50;   Approach position 1 and wait until axis is
                     stationary
N070 WAIT 1000;
N080 GOW A400 V50;   Position 2
N090 WAIT 1000;
N100 GOW A500 V50;   Position 3
N110 WAIT 1000;
N120 GOW A200 V150;  Return to initial position
N130 JMP N050;
END
```

[1)] Wait command is not necessary because program is not continued until after completion of reference run.

Example 3: Relative positioning

In example 2, the axis is always moved the same distance, which suggests a solution with relative positioning. A counter always indicates the current position.

```
%P02  (Example 2)
N010  GO 0;               Reference run
N030  GOW A200 V50;       Approach initial position and wait
N040  SET C00=0;          Set counter =0
N050  WAIT 1000;
N060  GOW R100 V50;       Approach next position
N070  SET C00+1;          Count with position counter
N080  WAIT 1000;
N090  JMP (C00<3) N060;   Position 3 not yet reached
N100  GOW A200 V150;      Return to initial position
N110  JMP N040
END
```

The solution is even simpler and more elegant if one does without the counter and the comparison is made with the set position:

```
%P02  (Example 2)
N010  GO 0;               Reference run
N030  GOW A200 V50;       Approach initial position and wait
N040  WAIT 1000;
N050  GOW R100 V50;       Approach next position
N060  WAIT 1000;
N070  JMP (SP<500) N050;  Position 3 not yet reached 1)
N080  GOW A200 V150;      Return to initial position
N090  JMP N040
END
```

The comparison is made with the set position because a comparison to the exact increment is problematic. The condition is not fulfilled unless the exact increment is reached.

Example 4: Sequential program

POSMOD1 is used here as a fully programmable sequence control system for a rotational speed profile.

An infinite conveyor belt is operated at two speeds. When a target position (≥ 10000) is reached, the belt is to be stopped. The cycle is repeated with a new start input.

Input	IP13	= START	(temporarily)
	IP14	= STOP	
	IP15	= speed 2	(at high level)
Output:	OP10	= target position reached	

The subprogram technique is used to keep the structure understandable. The main program carries out only the initialisation and calls up the subprograms 1 to 3 consecutively in an infinite cycle.

Main program:

```
%P00  (sequence)
N010  SET 0;              Set current position as home position
N020  SET M00=1;          Marker =1: Axis should not be started
N030  JMP P01;            Scan SP inputs
N040  JMP P02;            Start SP axis
N060  JMP P03;            SP position comparison
N070  JMP N030;           Repeat
END
```

Subprogram 1: Scan inputs

```
%P01  (scan inputs)
N010  JMP (IP14=0) N030;  Input STOP
N015  STOP 0
N020  JMP N010;           As long as STOP is not reset
N030  JMP (IP13=0) END;   No start
N040  SET M00=0;          Start was given, marker set =0
N050  SET H00=50;         Set speed to 1
N060  JMP (IP15=0) END;   Speed 1 selected
N070  SET H00=50;         Speed 2 selected and set
END
```

Subprogram 2: Start axis

```
%P02  (start axis)
N010  JMP (M00=0) END
N020  GO V H00;          Start axis
N030  SET M00=1;         Start is recognised, set back marker
END
```

Subprogram 3: Position comparison

```
%P03  (position comparison)
N030  JMP (IP>9999) N050;If target position is reached
N035  SET OP10=0;        Otherwise set output back
N040  JMP END;           continue
N050  STOP B;            Target reached, brake
N060  SET M00=1;         Set marker back
N070  SET OP10=1;        Set output
END
```

Example 5: Reference run

This example shows how to set a reference run oneself, without using any of the prescribed types.

```
%P00  (reference run)
N005  SET 0; Set current home position position temporarily
N006  SET OV=1;          Switch on override function
N010  JMP (M00=1) N140;  M00=1 => Reference run completed
N020  JMP (IP17=1) N100; IP17=1 Drive on reference cam
N030  GO V H51;          Infinite run in positive direction of
                         encoder counting
N040  JMP (IP17=0) N040; Ref. cam found
N050  STOP B;            Stop axis
N100  GO W R-20 V50;     Move drive from reference cam, rel 20
                         distance units
N110  GO V+10;           Search for edge of reference cam at slow
                         speed
N120  JMP (IP17=0) N120; Wait for edge
N130  SET 0;             Set home position
N140  STOP B;            Stop axis
END
```

9 Program and data management commands

These commands can be transmitted for operations over the RS485 interface or the CAN bus on the POSMOD1. LuPos also uses this instruction set in manual mode.

Protocol layout

For LustBus	→	Chapter 10.1
For CAN bus	→	Chapter 10.2

9.1 Transferring and calling up programs and program blocks

Transfer of a program (write to axis)

%Pyy (a)	Program header:	yy	= Program number (2-digit, 00 ... 99)
		a	= program name (max. 20 characters)
Ny ...	Program block:	y	= block number (0 ... 999)
"		...	= command
"			
END	End ID		

Comment: If an already existing program is to be overwritten, this must be erased first.

The entry of a program name can be passed over. The parentheses () are absolutely necessary.

Calling up a program (read from axis)

%RD Pyy	(RD = Read)	yy = Program number (2-digit, 00 ... 99)

Acknowledgement:	if program is available	see "Transfer of a Program"
	if program is not available	Error message

Transfer of a program block

%Pyy Ny ...	yy	= Program number (2-digit, 00 ... 99)
	y	= block number (0 ... 999)

If the program in question does not exist, an error message is generated.
If the program in question and the block number already exist, the existing program block is overwritten, otherwise, the "new" block is inserted into the existing program.

Calling up a program block

%RD Pyy Ny	yy	= Program number (2-digit, 00 ... 99)
	y	= block number (0 ... 999)

Acknowledgement:	if program and block number are not available	%Pyy Ny ...
	if program and block number are not available	Error message

9.2 Deleting programs and program blocks

Deleting a program

%CL Pyy (CL = Clear)	yy	= Program number (2-digit, 00 ... 99)

Deleting all programs

%CL PXX (CL = Clear)

Deleting a program block

%CL Pyy Ny	yy	= Program number (2-digit, 00 ... 99)
	y	= block number (0 ... 999)

Acknowledgement:	if program and block number are available	ACK
	if program and block number are not available	Error message

9.3 Calling up the table of contents

%DIR (DIR = Directory)

Acknowledgement:
- if available:
  Program numbers, names, and number of programmed blocks
- Version number of the company software
- Total number of programmed blocks and
- Number of blocks still available

Example:
P01 (example 1) N20
P05 (example 5) N35
V1.01 USED: 55 FREE: 2945

9.4 Storing data in flash EPROM

This command saves the user data (machine parameters, table positions, programs and the variable values H50...H90) located in working memory in the non-volatile memory (flash EPROM). The procedure takes several seconds.

%SAV (SAV = Save)

9.5 Transferring and calling up machine parameters

These parameters contain data concerning the machine and are not usually changed. They are stored in non-volatile memory and are read in after switching on.

Transfer of machine parameters

They can be transferred completely, partially or individually.

%Kxx:a

xx = Index of the machine parameter
a = Value of the machine parameter

Calling up all machine parameters

%RD KXX (RD = Read)

Acknowledgement: See "Transfer of Machine Parameters"

Calling up one machine parameter

%RD Kxx (RD = Read)

Acknowledgement: Kxx:a xx = Index of the machine parameter
a = Value of the machine parameter

9.5.1 Calling up parameters for sin² acceleration

A command is available to help with setting parameters for sin²-shaped acceleration, which can be issued from LuPos in the manual mode menu.

%RDS Returns various parameters which were defined in the calculation of the sin²-shaped acceleration parabolas.

It can be especially useful for the user to use this function to determine the distance which was travelled during sin²-shaped acceleration or braking phases (e.g. = 2 x distance[AccDataP.MaxIndex] + distance[linear phase]).

Acknowledgement:

Accelerate		Brake		Meaning
AccDataP.MaxIndex:	x	DecDataP.MaxIndex:	y	Number of calculated parabola points for approaching/braking (+)
AccDataN.MaxIndex:	x	DecDataN.MaxIndex:	y	Number of calculated parabola points for approaching/braking (-)
Vel [AccDataP.MaxIndex]:	x	Vel [DecDataP.MaxIndex]:	y	Current speed at end of approach or brake parabola (+)[1)]
Vel [AccDataN.MaxIndex]:	x	Vel [DecDataN.MaxIndex]:	y	Current speed at end of approach or brake parabola (-)[1)]
Distance [AccDataP.MaxIndex]:	x	Distance [DecDataP.MaxIndex]:	y	Distance travelled up to end of approach or brake parabola (+)[2)]
Distance [AccDataN.MaxIndex]:	x	Distance [DecDataN.MaxIndex]:	y	Distance travelled up to end of approach or brake parabola (-)[2)]
DeltaV lin.accel. pos:	x	DeltaV lin. braking pos:	y	Speed change per 5 ms due to linear accel. parameter (+)[1)]
DeltaV lin.accel. neg:	x	DeltaV lin. braking neg:	y	Speed change per 5 ms due to linear accel. parameter (-)[1)]
DeltaV sin accel. pos:	x	DeltaV sin braking pos:	y	Speed change per 5 ms at highest point of the approach or brake parabola (+)[1)]
DeltaV sin accel. neg:	x	DeltaV sin braking neg:	y	Speed change per 5 ms at highest point of the approach or brake parabola (-)[1)]

Positive direction		Negative direction		Meaning
K17 factor:	x	K18 factor:	y	Current value of adjustment factors for the acceleration parameters K17 to K24 (with SET commands from 0 ... 100%)
K19 factor:	x	K20 factor:	y	
K21 factor:	x	K22 factor:	y	
K23 factor:	x	K24 factor:	y	
Current accel. type pos.:	x	Current accel. type pos.:	y	Current accel. type in positive or negative direction (x, y= linear/sin)

x, y Values of parameters:
(+) in positive direction
(-) in negative direction
1) in speed units (K12)
2) in distance units (K10/K11)

9.5.2 Coded representation of inputs and outputs.

The machine parameters for configuration of inputs and outputs are represented in code. This applies to calling up and transferring as well as printing out machine parameters.

The first byte of the machine parameter determines the number of the port.

00 = POSMOD1 I/O
01 ... 04 = I/O of CAN modules

The second byte defines the number(s) of the inputs or outputs addressed in hexadecimal code.

Example:
Which input is adressed when a machine parameter takes the value "0108"?

It follows from the first byte, "01", that it is an input of the first CAN module (node module). Which input of this module is activated can be determined by the binary representation of the second byte, "08":

Position	7	6	5	4	3	2	1	0	
08_{Hex} =	0	0	0	0	1	0	0	0	$_{Bin}$

Because bit 3 is set, input 3 of the IC13 CAN node module is adressed.

If a machine parameter only activates one input or output, all possible combinations of the second byte can be taken from the following table:

Parameter value	Assigned input/output
xx00	Function is assigned to no I/O (is available in program)
xx01	Ixx0
xx02	Ixx1
xx04	Ixx2
xx08	Ixx3
xx10	Ixx4
xx20	Ixx5
xx40	Ixx6
xx80	Ixx7

The advantage of hexadecimal coding is that more than one input or output can be activated with a single byte. The parameter value of "0006" can, for example, assign the inputs IP11 and IP12 to a function (06_{Hex} = 0000 0110 $_{Bin}$).

If several inputs or outputs are to be assigned to a function, the corresponding value of the machine parameter is easily determined using the example above.

9.6 Transferring and calling up variables

This is used to store position, speed, counter and timer values which are needed again later by user programs.
The variable values from H00...H49 are set to 0 at start-up, the higher values H50 ...H99 are set to the values stored in the non-volatile memory.
Values can be loaded at any point in time over the serial interface or by particular commands within a program, or in manual mode. In the first case, the values can be transferred completely, partially or individually.

Transfer of variables

%Hxx:x

xx = index of the variables
x = value of the variables

Calling up the entire variable memory

%RD HXX (RD = Read)

Acknowledgement: See "Transfer of Variables"

Calling up one variable

%RD Hxx (RD = Read)

Acknowledgement: Hxx:x xx = index of the variables
x = value of the variables

9.7 Transferring and calling up table positions

Position values can be stored here on which one can position in user programs by selecting with inputs and the GOT... commands.
The table positions can be transferred completely, partially or individually.
After switching on, the table positions are set to the values stored in flash-EPROM.
Table positions can be loaded at any point in time over the serial interface or by particular commands within a program, or in manual mode. In the first case, the values can be transferred completely, partially or individually.

Transfer of table positions

%Txx:x

xx = index of the table positions
x = Value of the table position

Calling up the variable memory

%RD TXX (RD = Read)

Acknowledgement: See "Transfer of Table Positions"

Calling up a table position

%RD Txx (RD = Read)

Acknowledgement: Txx:x xx = index of the table positions
x = value of the table positions

9.8 Transferring and calling up markers

Markers are variables which can only take two possible conditions: 0 or 1. They are set in the program with the command SET Mxx =..., and called up with the command JMP (Mxx =...).

After switching on, all markers are set to 0. Markers maintain their values until they are overwritten by a new command, i.e. markers set in one program can be called up in another program (e.g. a subprogram).

Markers can be transferred and called up at any point in time over the serial interface or by particular commands within a program, or in manual mode. In the first case, the values can be transferred completely, partially or individually.

Transfer of markers

%Mxx:a

xx = index of the marker
a = value of the marker

Calling up all markers

%RD MXX (RD = Read)

Acknowledgement:

%M00:a	M01:a	M02:a	...	M09:a
M10:a	M11:a	M12:a	...	M19:a
M20:a	M21:a	M22:a	...	M29:a
:	:	:	:	:
M90:a	M91:a	M92:a	...	M99:a

a = value of the marker (0 or 1)

Calling up a marker

%RD Mxx (RD = Read)

Acknowledgement: %Mxx:a xx = index of the marker
a = value of the marker (0 or 1)

9.9 Calling up status

Calling up the axis status

The command "?" is given constantly by the LuPos user interface within the "Manual mode", "Reference run" and "Status display" menus. It is for cyclical display of returned values.

Command: ?

Acknowledgement:

G ...	Actual position	[distance unit]	
H ...	Set position	[distance unit]	
O ...	Contouring gap	[Increments]	
I ...	SR-/PE-Status	Hexadecimal, 4-digit	
J ...	Override	[%]	
K ...	local inputs	Hexadecimal. 2-digit	
L ...	local outputs	Hexadecimal. 2-digit	
M ...	consecutive program numbers	decimal	
N ...	consecutive block numbers	decimal	
P ...	Hardware limit switch	hexadecimal, 1-digit;	Bit 0 : Positive limit switch Bit 1 : Negative limit switch

Calling up the CAN I/O status:

The command "%RDCIO" is given constantly in the LuPos user software within the "Status display" menu after selection of ”CAN-I/O”. It is for cyclical display of the status.

Command: %RDCIO

Acknowledgement:
No CAN module available: -
Otherwise: Status of the CAN module (I = input, O = output) in the order of the sequence (hexadecimal).

Example: I29 OB4 O5F

This corresponds

Module No.:	Bits:
IC0 =	00101001
OC0 =	10110100
OC1 =	01011111

to:

10 Operations with LUSTBUS or CAN-Bus

10.1 Operations with LUSTBUS (RS485)

Telegrams can be sent to the POSMOD1 option of the MC6000 servocontroller with the parameter number CODE = 49999.
There are two types of telegrams: the select and the inquiry telegram. With the select telegram, which is always sent first, data can be transferred to POSMOD1. Data can be called up from POSMOD1 with the inquiry telegram. Only one particular CODE string (49999) is intended for data exchange with POSMOD1.
If the MC6000 signals an error condition with NACK, the error is queried by the master computer with the read instruction %RDF.

10.1.1 SELECT telegram

Master computer

EOT	*@*	*STX*	*4*	*9*	*9*	*9*	*9*	*=*	*0 ... 80 Byte ASCII Data (use data)*	*ETX*	*BCC*

Data is sent to POSMOD1 or data to be called up is specified using the select telegram.

10.1.2 ENQUIRY telegram

Master computer:

EOT	*@*	*4*	*9*	*9*	*9*	*9*	*ENQ*

Data previously specified with the SELECT telegram is called up with the ENQUIRY telegram. The acknowledgement telegram finally sent by the MC6000 has the following structure:
Servocontroller:

@	*STX*	*4*	*9*	*9*	*9*	*9*	*=*	*0 ... 80 Byte ASCII Data (use data)*	*ETX*	*BCC*

10.1.3 Layout of use data

The use data consists of a **status byte** followed by the actual telegram data. The contents of the status byte are as follows:

Bit 0: 1 = call up telegram

This is a SELECT telegram which specifies the type of data to be called up (e.g. a status call-up from POSMOD1).

0 = data telegram

This is a telegram which contains no data call-up and which must only be acknowledged by the receiver with an ACK (e.g. a positioning command in manual mode).

Bit 1: is always 0

Bit 2: 1 = multiblock telegram

This is a telegram with data which is followed by still further telegrams after acknowledgement by the receiver with an ENQUIRY telegram (e.g. when transferring programs).

0 = individual telegrams

This is an individual telegram or the last telegram of a multiblock transmission which is not followed by further telegrams.

Bit 3 1 = ACK desired
0 = ACK not desired

Bit 4, 5 always = 1

Bit 6, 7 always = 0

Please take the use data from the instruction set description in the User Manual.

Example of status byte: (e.g. transfer of a positioning command to PosMod1)

Bit	7	6	5	4	3	2	1	0
Binary	**0**	**0**	**1**	**1**	**1**	**0**	**0**	**0**
Hexadecimal			**3**				**8**	**H**

10.1.4 Example telegrams

Transfer of a positioning command to PosMod1

Master computer:

EOT	*ADR*	*STX*	*4*	*9*	*9*	*9*	*9*	*=*	*38H*	*GOA0V1000*	*ETX*	*BCC*

Servocontroller:

ADR	*ACK*

or

ADR	*NAK*

Master computer:

EOT

Transfer of a program to PosMod1

Master computer:

EOT	*ADR*	*STX*	*4*	*9*	*9*	*9*	*9*	*=*	*3CH*	*%P10 (RPF)*	*ETX*	*BCC*

Servocontroller:

ADR	*ACK*

or

ADR	*NAK*

Master computer:

EOT

EOT	*ADR*	*STX*	*4*	*9*	*9*	*9*	*9*	*=*	*3CH*	*N10 GO 0*	*ETX*	*BCC*

Servocontroller:

ADR	*ACK*

or

ADR	*NAK*

Master computer:

EOT

EOT	*ADR*	*STX*	*4*	*9*	*9*	*9*	*9*	*=*	*38H*	*END*	*ETX*	*BCC*

Servocontroller:

ADR	*ACK*

or

ADR	*NAK*

Master computer:

EOT

Status call-up from PosMod1

Master computer:

EOT	ADR	STX	4	9	9	9	9	=	31H	?	ETX	BCC

Servocontroller:

ADR	ACK

or

ADR	NAK

Master computer:

EOT

EOT	ADR	4	9	9	9	9	ENQ

Servocontroller:

ADR	STX	4	9	9	9	9	=	30H	G10H10 ...	ETX	BCC

Master computer:

EOT

Program call-up from PosMod1

Master computer:

EOT	ADR	STX	4	9	9	9	9	=	31H	%RDP10	ETX	BCC

Servocontroller:

ADR	ACK

or

ADR	NAK

Master computer:

EOT

Master computer:

EOT	ADR	4	9	9	9	9	ENQ

Servocontroller:

ADR	STX	4	9	9	9	9	=	34H	%P10 (RPF)	ETX	BCC

Master computer:

EOT

Master computer:

EOT	ADR	4	9	9	9	9	ENQ

Servocontroller:

ADR	STX	4	9	9	9	9	=	34H	N10 GO 0	ETX	BCC

Master computer:

EOT

Master computer:

EOT	ADR	4	9	9	9	9	ENQ

Servocontroller:

ADR	STX	4	9	9	9	9	=	30H	END	ETX	BCC

Master computer:

EOT

10.2 Operations with CAN-Bus

In servocontroller software version V150.3 and above and in PosMod1 company software version V1.03 and above, a CAN bus interface is supported. This has a manufacturer-specific protocol.

The present CAN bus interface has the same functionality as the RS485 interface, i.e. all functions listed in chapter 9 can be used.

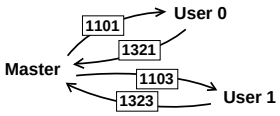
The standard servocontroller parameter channel is used with the CAN bus. The data to be transferred is simply transferred by the CAN bus as a use data block. The data corresponds to that of the RS485 (parameter number + interface command).

Note:
Direct control of the servocontroller over the control channel (basis identifier 661) of the CAN bus is not possible because PosMod1 adopts control of the servocontroller.
The cyclical status message (basis identifier 881) is usable.

10.2.1 Servocontroller parameter channel

Data flow direction: Master -> Servo

Parameters are transferred or queried on this ID. Every transmission of this ID is followed by a reply with the ID 1321.



Priority according to CAL	Basis ID	Data byte 0	Data byte 1	Data byte 2	Data byte 3+4+5+6	Data byte 7
5	1101	PARA_LO	PARA_HI	Type of transfer: "SEL" = 02 "ENQ" = 05	DATA	COUNTER

PARA_LO: Parameter number low byte
PARA_HI: Parameter number high byte
(here: PARA_LO + PARA_HI = 49999 Dec. = **4FC3 Hex**, code string for PosMod1)

Type of transfer:

00 =	List_End	Transfer end
02 =	SEL	Data transfer to servocontroller
05 =	ENQ	Data call-up to servocontroller
06 =	ENQ_POSMOD	Data call-up to PosMod1
07 =	SEL_POSMOD	Data transfer to PosMod1

DATA: 32-bit data (1st byte = see 10.1.3)
(at "list end" : running digital sum)

COUNTER: Block counter
(incremented at each transfer)

Data flow direction: SERVO -> Master

Priority according to CAL	Basis ID	Data byte 0	Data byte 1	Data byte 2	Data byte 3+4+5+6	Data byte 7
5	1321	PARA_LO	PARA_HI	SIO STATUS 0 = transfer OK BIT 0 = Power on 1 = SIO watchdog 2 = transfer type unknown 3 = reading unauthorised 4 = repeat action 5 = parameter unknown 6 = change unauthorised 7 = unauthorised value	DATA	Counter

PARA_LO: Low byte parameter number

PARA_HI: High byte parameter number

DATA: 32-bit data (at "List end" : running digital sum)

COUNTER: Block counter (incremented at each transfer of a new block within an array).

10.2.2 Examples

ID	Data	Telegram	Function
	% M 0	**SEL**	**Set marker 1**
1103	4F C3 07 38 25 4D 30 00		
			49999: SEL_POSMOD;
1323	4F C3 00 38 25 4D 30 00		Status byte (see chapter 10.1)
	1 : 1		%M01:1 (see chapter 9.8)
1103	4F C3 07 31 3A 31 00 01		Counter
1323	4F C3 00 31 3A 31 00 01		
	running digital sum		
1103	4F C3 00 09 1F 7C 30 02		
1323	4F C3 00 09 1F 7C 30 02		

Legend:

4F C3	CODE string 49999 (for POSMOD1)
4F C3 00	Type of transfer: 00 = List_End 06 = ENQ_POSMOD 07 = SEL_POSMOD

ID	Data	Telegram	Function
	% R D	**SEL**	**Query marker 1**
1103	4F C3 07 38 25 52 44 00		
			see ....
1323	4F C3 00 38 25 52 44 00		%RD M01 (see chapter 9.8)
			Counter
	M 0 1		
1103	4F C3 07 4D 30 31 00 01		
1323	4F C3 00 4D 30 31 00 01		
	running digital sum		
1103	4F C3 00 75 15 63 44 02		running digital sum
1323	4F C3 02 75 15 63 44 00		
	undefined	**ENQ**	**Reply marker 1**
1103	4F C3 06 13 01 00 6F 00		
1323	4F C3 00 30 25 4D 30 00		%M01:1 (see chapter 9.8)
	% M 0		
	undefined		
1103	4F C3 06 13 01 01 6F 00		
1323	4F C3 06 31 3A 31 00 01		
	1 : 1		
	undefined		
1103	4F C3 06 13 01 02 6F 00		
1323	4F C3 00 01 1F 7C 30 02		running digital sum
	running digital sum		

	! % R D	**SEL**	**Query variable H50**
1103	4F C3 07 21 25 52 44 00		
1323	4F C3 00 21 25 52 44 00		%RD H50 (see chapter 9.6)
	H 5 0		
1103	4F C3 07 48 35 30 00 01		
1323	4F C3 01 48 35 30 00 00		
	running digital sum		
1103	4F C3 00 69 10 62 44 02		
1323	4F C3 02 69 10 62 44 00		

ID	Data	Telegram	Function
	undefined	**ENQ**	**Reply variable H50**
1103	4F C3 06 12 01 00 F3 00		
1323	4F C3 00 30 25 48 35 00 % H 5		%H50:10000 (see chapter 9.6)
	undefined		
1103	4F C3 06 12 01 01 F3 00		
1323	4F C3 06 30 3A 31 30 01 0 : 1 0		
	undefined		
1103	4F C3 06 12 01 02 F3 00		
1323	4F C3 06 30 30 30 00 02 0 0 0		
	undefined		
1103	4F C3 06 12 01 03 F3 00		
1323	4F C3 00 30 2F 49 05 03 running digital sum		

Legend:

4F C3	CODE string 49999 (for POSMOD1)
4F C3 00	Type of transfer: 00 = List_End 06 = ENQ_POSMOD 07 = SEL_POSMOD

Appendix A Instruction set overview

Command	Operand 1	Op. 2	Comment
Jump commands / subprogram call-ups			
JMP	Ny/END/Pyy		Unconditional jump / subprogram call-up
	(IP < > x, Hxx)	Ny/END/Pyy [1]	Actual position
	(SP < = > x, Hxx)	Ny/END/Pyy [1]	Set position
	(PW = 0/1)	Ny/END/Pyy [1]	Actual position in position window (PW=1)
	(ST = 0/1)	Ny/END/Pyy	Set position = target position (ST=1)
	(Ippi = 0/1)	Ny/END/Pyy	Status of the input
	(Mxx = 0/1)	Ny/END/Pyy	Status of the marker
	(M[Cxx] = 0/1)	Ny/END/Pyy	Status of the marker (indexed)
	(Cxx < = > d)	Ny/END/Pyy	Counter reading
	(Hxx < = > z, Hyy)	Ny/END/Pyy	Quantity of the variable
	(H[Cxx] < = > z, Hyy)	Ny/END/Pyy	Quantity of the variable (indexed)
	(Zxx < > d, = 0)	Ny/END/Pyy	Timer reading
Setting commands			
SET	Oppi = 0/1, Mxx		Set output directly or with marker
	Oppi = (A | & B)		Set output over link (A, B = Ippi or Mxx; | = OR (ASCII 124) ; & = AND)
	Opp = Pmm		Set output port with mask (Pmm=00...FF Hex)
	Opp = Cxx		Set output port with counter reading (Pmm=00...FF Hex)
	Mxx = 0/1, Myy, -Myy		Set marker; invert
	M[Cxx] =0/1, Myy, -Myy		Set marker (indexed); invert
	Mxx = (A | & B)		Set marker over link (A = Ippi; B = Ippi or Mxx; | = OR (ASCII 124) ; & = AND)
	Hxx = z, Hyy, H[Cyy], Cyy		Set variable
	H[Cxx] = z, Hyy, H[Cyy], Cyy		Set variable (indexed)
	Hxx + - * : z, Hyy		Calculate variable
	H[Cxx] + - * : z, Hyy		Calculate variable (indexed)
	Hxx = IP, SP, OV		Set variable
	H[Cxx] = IP, SP, OV		Set variable (indexed)
	Hxx = Txx		Set variable with table position
	H[Cxx] = Txx		Set variable with table position (indexed)
	Cxx = d, Hyy, Cyy		Set counter
	Cxx + - b, Hyy		Increment / decrement counter
	Cxx = Ipp		Set counter with input port
	Txx = x, Hxx, IP, SP		Set table position
	Zxx = d, Hxx		Set timer
	OV = 0/1		Set override (OV=1: switch on)
	0		Transfer current position as reference point
	K15, K16 = 0/1, Hxx [2]		Set acceleration type (0=linear, 1=sin²) (K15= positive, K16= negative direction)
	K17...K24 = p, Hxx [2]		Set maximum acceleration value (p=1...100%)

[1] Execution time = 5 ... 10 ms

[2] Execution time ≤ 90 ms

Command	Operand 1	Op. 2	Comment
Positioning and traversing commands			
GO	Ax, A Hxx	Vy, Hyy	Absolute position, speed (with continuation)
	Rx, R Hxx	Vy, Hyy	Relative distance, speed (with cont.)
GOTA	Vy, Hyy		Position from table, speed (with cont.)
GOTR	Vy, Hyy		Distance from table, speed (with cont.)
GOW	Ax, A Hxx	Vy, Hyy	Absolute position, speed (without continuation)
	Rx, R Hxx	Vy, Hyy	Relative distance, speed (without cont.)
GOTWA	Vy, Hyy		Position from table, speed (without cont.)
GOTWR	Vy, Hyy		Distance from table, speed (without cont.)
GO	0		Execute reference run (in accordance with K70)
	V + - y		Infinite traversing (directly)
	V Hxx		Infinite traversing (with variable)
STOP	B		Brake at programmed deceleration (linear or sin^2)
	M		Brake at maximum deceleration (linear, K19 / K20)
	0		Brake at maximum deceleration Then switch off the position control system
Wait commands and storing			
WAIT	b, Hxx		Waiting time in ms (1...65535)
	PW		Wait until axis is in position window (actual position)
	ST		Wait until axis is in target position (set position)
	(Ippi = 0/1)		Wait until input has reached level
SAVE			Secure data (from RAM into flash EPROM) [1)]

1) Execution time ca. 6 s

Symbol	Meaning	Value range	
Cxx	Counter index	00 ... 99	
Hxx, Hyy	Variable index	00 ... 99	
Kxx	Machine parameters	00 ... 99	
Mxx, Myy	Marker index	00 ... 99	
Ny	Block number	000 ... 999	
Pmm	Mask	00 ... FF Hex	(8 bit)
Pyy	Program number	00 ... 99	
Txx	Table positions	00 ... 15	
Zxx	Timer index	00 ... 07	
a	Value of a marker	0 / 1	
b	Value (auxiliary quantity for loading / calculation)	1 ... 65535	(16 bit)
d	Status of a timer or counter	0 ... 65535	(16 bit)
x, y, z	Numerical value (position, speed, variables or table positions)	-2.147.483.647 ... +2.147.483.647	($-2{,}1*10^9$ ... $+2{,}1*10^9$)

Appendix B Error messages

If an error occurs, the output OP13 is set, if it has been programmed as a fault signal. The KEYPAD shows the error location number in the upper left corner of the display.

The program in progress is halted if an error occurs. After the error has been solved, the program must be restarted.

Note:
When using POSMOD1 it is a good idea to reset errors with the AUTOMATIC input of POSMOD1. This also resets servocontroller errors and the motor is usually switched off (depending on the severity of the error).

a) Servocontroller error messages in connection with POSMOD1:

Error location No.	Meaning
200	POSMOD1 not recognized (circuit board not attached correctly or incorrect software)
201	POSMOD1 not ready for operation (probably defective)
202	Time-out during initialization handshake (probably defective)
203	Servocontroller does not support current POSMOD1 SW version (update servocontroller software)
204	New position set point not delivered on time by POSMOD1 (contact service department if error reoccurs).

b) Fatal errors:
These errors can only be reset by cutting the power supply.

Error location No.	Meaning
211	Hardware limit switches reversed
212	Time-out dduring boot synchronisation
213	Watchdog is activated

c) Non-fatal errors (numbers above 220):
These errors can be reset over the serial interface by:
- Changing level on "Automatic" input
- Activating reference run with "Start" input in manual mode.

Caution!
It is also possible to reset an error over the ENPO input (power stage release). Hoowever, this switches off the control system (not authorized in all applications)!

Error location No.	Meaning
220	Positive hardware limit switch approached [1)]
221	Negative hardware limit switch approached [1)]
222	Positive software limit switch approached [1)]
223	Negative software limit switch approached [1)]
224	Reference point not defined
225	Addressed hardware not present
226	Selected program not available
227	Jump to non-existent program block
228	Called-up program is not available
229	Target position outside of the traversing range

230	Division by zero
231	Max. nesting depth exceeded
232	Time-out in manual mode
233	Target position not reached
234	Feed hold missing
235	CAN time-out
236	Index overrun (indexed addressing)
237	Can reception cache full
238	CAN controller overflow
239	CAN controller error
240	Max. speed (K14) > servo speed limitation
241	CKM100 error
242	Controller release missing
243	Error message from servocontroller
244	PosMod output error (short-circuit, overheat or faulty supply)
245	Unauthorized command during axis motion

[1)] Move from limit switch in jog mode (inputs "jog +/-")

Appendix C Technical data

The most important technical data of PosMod1 are summarized here. See chapter 2.3 for a more exact description.

Cycle time of the position control system	500	µs	Fine interpolation
Position set point input from PosMod1	5	ms	
Resolution on the motor shaft	16	Bit	= 360°/65.536
Stationary position error on the motor shaft [1)]	≥ 0,011	°	Optical encoders (sin/cos)
	≥ 0,4 /p	°	Resolvers
Position ripple (f = 2 kHz) [1)]	≥ 0,006	°	Optical encoders (sin/cos)
	≥ 0,2 /p	°	Resolvers
PosMod1 internal cycle time	1 ... 3	ms	typical
	6	ms	maximum
Block-to-block execution time	1 ... 3	ms	typical
Read input / set output (PosMod1)	1 ... 3	ms	typical
Read input / set output (CKM100)	7 ... 10	ms	typical
Read status:	5 ... 10	ms	typical
SET K15 ... K24	≤ 90	ms	
Time between selection of automatic mode and subsequent start command	≥ 20	ms	

[1)] The numbers given are guidelines which are dependant on various limiting conditions. Check the values individually if necessary!

Appendix D Table of key words

—A—

Automatic..... 3-4
Automatic mode..... 2-1
Axis in position..... 3-7

—B—

Baud rate..... 4-2
Braking 7-14

—C—

Command syntax..... 7-1
Commisioning 1-4
Commissioning 1-3
Configuration of the CKM100 3-16
Creep feed..... 5-16

—D—

Deleting programs..... 9-2
Diagnosis menu 4-6
Distance resolution..... 5-5

—E—

Electrical connections.... 3-1
Error messages 10-2
Error report..... 4-7

—F—

Fault signal (output) 3-7; 5-3
Feed hold (input)..... 3-4

—H—

Hardware limit switch 3-5
Home position..... .. 3-7; 5-16; 7-10; 8-2; 8-5

—I—

Infinite moving..... 7-14
Inputs ...3-8; 3-11; 3-14; 5-1
Instruction set..... 7-1

—J—

Jog +/-..... 3-5
Jog mode 2-1
Jump commands 7-3

—K—

Key assignment..... 4-8
KEYPAD..... 2-2

—L—

Lag distance..... 2-6
Lag distance tolerance..... 5-17

—M—

Manual mode..... 3-4; 3-5
Markers..... 6-1
Maximum sampling time 2-6
Maximum speed 5-9

—N—

Node address.....3-16; 5-18

—O—

Outputs 3-6; 3-8; 3-11; 3-14; 5-1
Override 3-12; 7-11

—P—

Parameters for the servocontroller 2-2
Position window.....5-17
Printing out machine parameters 9-4
Printout with machine parameter screen..... 8-3
Program editor..... 4-7
Program end 3-7
Program examples 8-2
Program selection..... 3-5
Programming menu..... 4-3

—R—

Read-in hold (input)..... 3-4
Reference cam..... 3-6
Reference point defined 3-7
Reference run 2-1; 4-4; 5-19; 7-14
Resolution..... 5-6

—S—

Sampling time 2-6
Setting commands 7-7
Setup menu 4-4
Setup mode 2-1
Sin²-shaped acceleration5-12
Software limit switch.... 5-16
Start input 3-4
Status byte..... 10-1
Status display..... 4-6
Store data 4-4
Subprograms 7-3

—T—

Table index 3-5
Table position..... 6-2
Terminal module EKL300 3-9
Time behaviour 2-5
Timers 7-11
Traversing commands. 7-12

—V—

Variables..... 6-1

—W—

Wiring 3-8; 3-12; 3-19

—Z—

Zero offset 5-16

Lust Antriebstechnik GmbH * Gewerbestr. 5-9 * D-35633 Lahnau * Germany * Phone +49 64 41/966 -0 * Fax +49 64 41/966 -137