LinMot®

Release 1.3

(Includes Release 1.3.16)
Supplement to V1.0 User Manual
08/25/2008

Note
The information in this documentation reflects the stage of development at the time of press and is therefore without obligation.
NTI Ltd reserves itself the right to make changes at any time and without notice to reflect further technical advance or product improvement. Please refer to the latest edition of our "General business terms"

Version 1.19/ August, 25th 2008

1. *LinMot®* Software Innovations

1.1 Overview

For the Release 1.2/1.3 various extensions have been made to the PC software *LinMot®* Talk and to the firmware that runs on the *LinMot®* servo controller. This handbook describes all new functions and explains differences to Release 1.1. With all alterations care has been taken to preserve compatibility with Releases 1.0 and 1.1.

The following list provides an overview of the new functions. All innovations are described in detail in separate sections.

New functions in Release 1.3.16

- The version 3 servo controllers (E1001, E2001, E4001, E2031 and E4031) are supported. These controllers are also available with an integrated master encoder interface (no add on module needed anymore).
- The high performance motors P0x-23F-HP and P0x-37-HP are supported.
- The high clearance sliders for the motor combinations PS0x-37 with PL01-19, PS0x-37-HP with PL01-19, PS0x-37F with PL01-19 and PS0x-48 with PL01-27 are supported.

New functions in Release 1.3.15

- The motor series P0x-48 are supported with improved position tables.
- The curve wizard type “Limited Jerk” has now a onw parameter for the jerk definition. See chapter 1.5 “Limited jerk motion profiles”.

New functions in Release 1.3.14

- The motor series P0x-48 is supported.
- Noise dead band filter added when motor stands still. See chapter “Control parameters” on page 80.
- CANopen interface added. CANopen controllers are available as independant product types like DeviceNet controllers.
- Commands GotoPositionFromActualPosition and SetDemandPositionToActualPosition added in MT, ASCII, DP, DN and CO (CANopen) interface.
- ASCII: The command !EX (read state flags) contains now as well the emergency stop substate information.

New functions in Release 1.3.12

- On login *LinMot®* Talk checks the release consistency between PC and controller software.
- The serial link RS485 supports for both protocols RSTalk and ASCII RS485 the half and full duplex modes.

New functions in Release 1.3.11

- The ASCII command set has been expanded with commands for starting curves or cams from the actual position and commands for a setup service (reading and writing memory words) are added. See chapter 8.3 “Commands overview”.
- MT Commands for starting curves or cams from the actual position are added. See chapter 2.3 “State Table”.
- Profibus DP contains now modules for Error and Warning. See chapter 3.7 “Data module overview”.
- In *LinMot®* Talk the oscilloscope can display saved shots from a file even in the offline mode.

New functions in Release 1.3.10

- For setting up the motor parameters there is a new **Tuning Tool** which helps setting up the correct values for the feed forward **parameters** (such as Current Offset, FF_Acceleration and FF_Deceleration) according to the motor type, load mass, friction, etc. A detailed description of the tuning tool is located in chapter 7.7 “The Tuning Tool”.
- Shots taken with the *LinMot®* Talk’s built in oscilloscope can now be saved and recalled. This makes it very easy to exchange shots. See chapter 1.2 “Saving Oscilloscope Shots”.
- The **error inspector** displays additionally **controller state** and the **Logged Warnings**. This means all warnings which have occurred since the last run setup will be stored and displayed. This feature can be very helpful for commissioning machines. See chapter 1.3 “Logged Warnings”.
- Release 1.3.10 supports the **Gantry** mode which makes it easy for the user to initialize and command axis, where two motors work together but are mechanically linked in a weak way both of the two motors have to be position controlled. The gantry mode is available in combination with external sensors and booster motors. See chapter 5.2 “Master/Gantry operation”.
- Long stroke motors can now be run over the whole stroke without redefining the motor position or moving the home position. Therefore the sensor configuration mode **Internal Sensor 40µm** is added. By selection of this item, one position increment will be 40 µm which allows to cover **2520mm**. See chapter “Position sensing” on page 72.
- For some applications it is helpful, if all motor start their initialization (homing) at the same time. Therefore the global switch **Init Together** was added. See chapter 6.2 “Global parameters”.
- To open up the versatility of the position sensor types, **A/B sensors** are supported on the **master encoder module**. Up to two A/B sensors can be linked with any motor. See chapter 4.2 “A/B sensors”.
- Release 1.3.10 provides the **master encoder** functionality (see Addendum Master Encoder) in combination with the MT (Multitrigger), DP (Profibus) or DN (DeviceNet) interface. For each interface, there are added master encoder specific command for changing recipe, initiating cams or unlocking cam mode.
- The configuration and debugging protocol **RS-Talk** is supported on both links **RS232** and **RS485**. The two links are handled completely separately, thus it is even possible to communicate over the two links at the same time.

New Functions in Release 1.3.9

- Release 1.3.9 supports a new curve type **Limited Jerk** in the **Curve Creator**. For this type the maximal speed and maximal acceleration are configurable like for the **Point To Point** type. The acceleration (and thus the force) will not jump but change with a ramp.
- The curve type **Ramp** is not supported anymore. It can be replaced easily by the type **Manual** with two curve points and the time. When importing curve files in the **Curve Inspector** the curve type **Ramp** will be automatically converted to the **Manual** type.

New functions in Release 1.3

- Parallel (master/booster) operation of motors. This enables the force available for a movement to be increased in simple fashion.

- Support for external sensors, which make possible substantially higher accuracy. Thus positioning to an accuracy of 10µm is possible with the appropriate magnetic tape. The external sensors are connected to motor channels not in use.
- The *LinMot®* firmware supports the new PROFIBUS servo controller: E130-DP, E230-DP, E430-DP, E1030-DP, E2030-DP and E4030-DP.
- With the new PC software both servo controller with Release 1.2 and servo controller with the new Release 1.3 can be parameterize and operated.
- The **DISABLE** state is now signalled by the LEDs **Stat A** and **Stat B** flashing briefly twice. It is now distinguished unmistakely from the **STOP** state. The complete state diagram with all LED codes is shown in Fig. 1-5, operating states, on page 6.
- The multitrigger servo controller support four new commands. With **Set PID**, **Set FF**, **Set Cur. Offset** and **Set CP** the controller adjustment and the motion profile properties may be altered during operation (on the fly).
- New ASCII commands for starting movements with the help of trigger impulse and for writing and reading various parameters. A full description may be found in the section on *LinMot®* ASCII protocol on page 86.
- The speed, amplitude and position offset of the motion profiles can be set at run time with the ASCII protocol and the PROFIBUS.

New functions in Release 1.2

- Support of a new motion profile type for producing minimal jerk motion profiles. With this motion profile type very simple position profiles can be produced having minimal jerking.
- The so-called **Package Installer** facilitates equipping the servo controller to the latest release. With only few mouse clicks the servo controller programmed with release 1.0 or 1.1 can be equipped for release 1.2.
- Extended controller: The internal position controller has been optimized and given new functions. Through these extensions the controller can be better adapted to the purpose in demanding servo applications.
- Implementation of an ASCII protocol for the RS232 and RS485 interfaces. With this protocol very simple applications may be programmed, controlling several *LinMot®* motors via RS232 or RS485 interface. Via the protocol, predefined position profiles may be started and any target positions moved into. The new ASCII protocol is described in the section on the *LinMot®*ASCII protocol on page 86 of this handbook.
- Support of the operation of *LinMot®* P01-23x160 motors with the servo controller of the E1000 series. The motors are connected with an adapter cable.
- Importing motor configuration data from different servo controller is now assisted. If for example a motor configuration has been stored on E100 servo controller, this can be loaded without problems onto E200, E400, E1000, E2000 or E4000 servo controller.
- The multitrigger servo controller support two new commands. With the command **Redefine Position** the actual position can be redefined. With **Set Current** the maximum current and the force of a motor can be altered.
- New in the **Error Inspector** is a function for displaying the current I/O values.With the help of this I/O status display, problems during commissioning can be overcome efficiently.

- Support for the “big type” adjustment in Windows 95/NT. Users of large monitors can new use the “big type” setting. This ensures that all the displays on the screen remain legible, even when using 19” or 21” monitors.

1.2 Saving Oscilloscope Shots

The oscilloscope supports now saving and recalling shots. With the “Save Configuration” button a complete oscilloscope configuration with the sampled data included can be saved as a “.ose” file type. When Opening the saved configuration the stored data will be displayed on the screen.

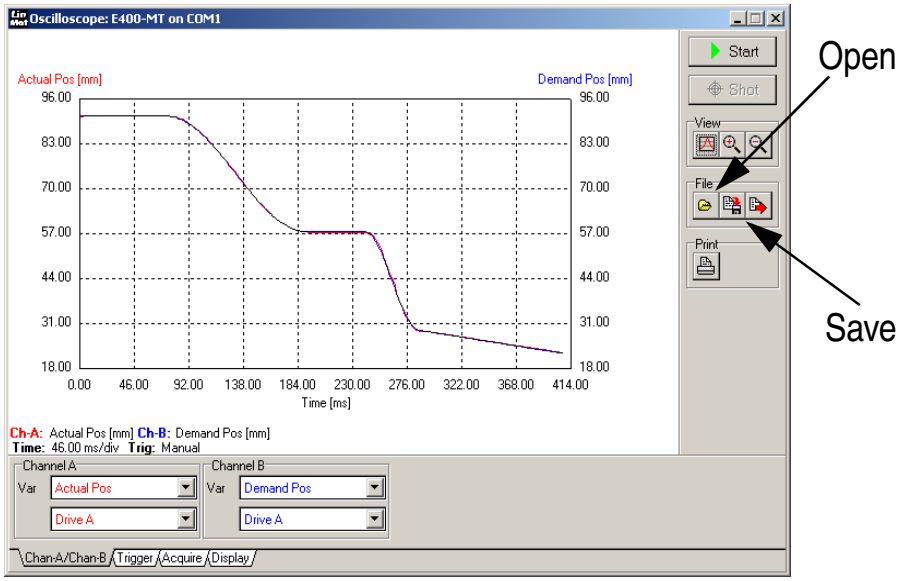


Figure 1-1: Saving or opening oscilloscope shots

1.3 Logged Warnings

The servo controller stores all warnings occurred since the last run setup (entering in the RUN mode). These so called logged warnings are read out and displayed with the **Error Inspector** by clicking the **Warnings** button.

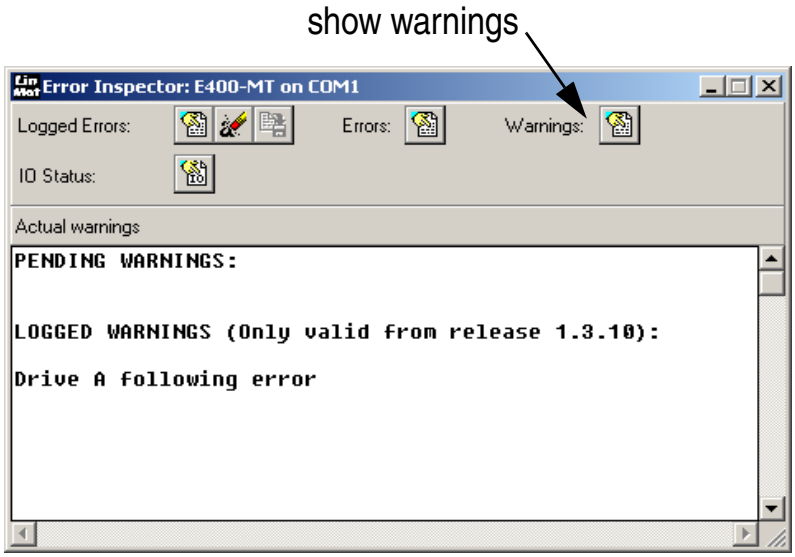


Figure 1-2: Displaying logged warnings

1.4 Minimal jerk motion profiles

To simplify the generation of such profiles the **Curve Editor** has been provided with a new tool - the wizard. This calculates the motion profile with the least possible jerking, taking into account the parameters start point, end point and desired traversing time. Figure 1-3, “Traverse, speed and acceleration of a minimal jerk motion” plots the traverse, speed and acceleration of such a profile.

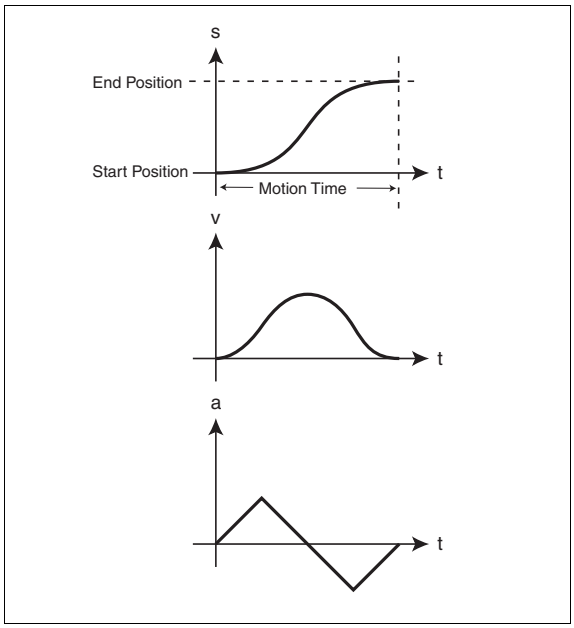


Figure 1-3: Traverse, speed and acceleration of a minimal jerk motion

Figure 1-4, “Typical minimal jerk motion profile” shows the **Curve Editor** with which a minimal jerk motion profile has been produced.

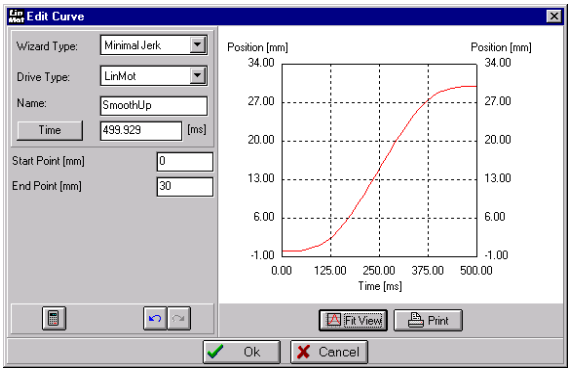


Figure 1-4: Typical minimal jerk motion profile

1.5 Limited jerk motion profiles

The wizard for **Limited Jerk** profiles has now an additional parameter for the jerk itself. The releases 1.3.9 to 1.3.14 have automatically assumed a triangular acceleration wave form.

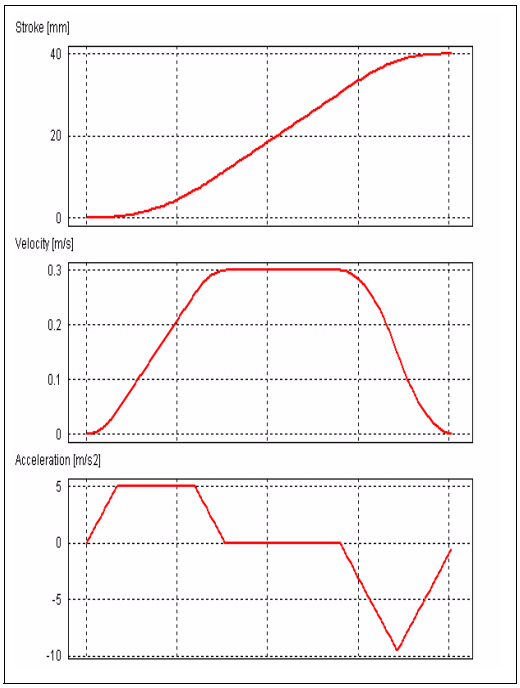


Figure 1-5: Traverse, speed and acceleration of a limited jerk motion

Figure 1-6, “Typical limited jerk motion profile”, on page 7 shows an example of a limited jerk profile generated in the curve editor. When importing curves of type limited jerk, which are generated in versions 1.3.9 to 1.3.14, they will be automatically converted into manual types.

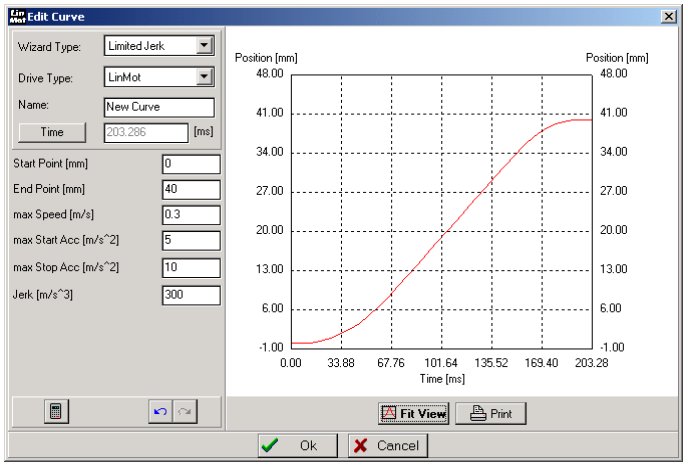


Figure 1-6: Typical limited jerk motion profile

1.6 Package installer

Equipping the servo controller for a new SW release has entailed loading many individual data files onto the servo controller. With the new **Package Installer** a release may now be loaded with a few mouse clicks. The installer itself ascertains which servo controller is involved and automatically loads the software for the particular version.

The **Package Installer** is started by selecting the menu option ‘Special’ -> ‘Install Package’. After this **Install** as User-ID and as password **NTI** must be entered in capitals. The installation file is in the **LinMot\Lin…\Firmware** directory.

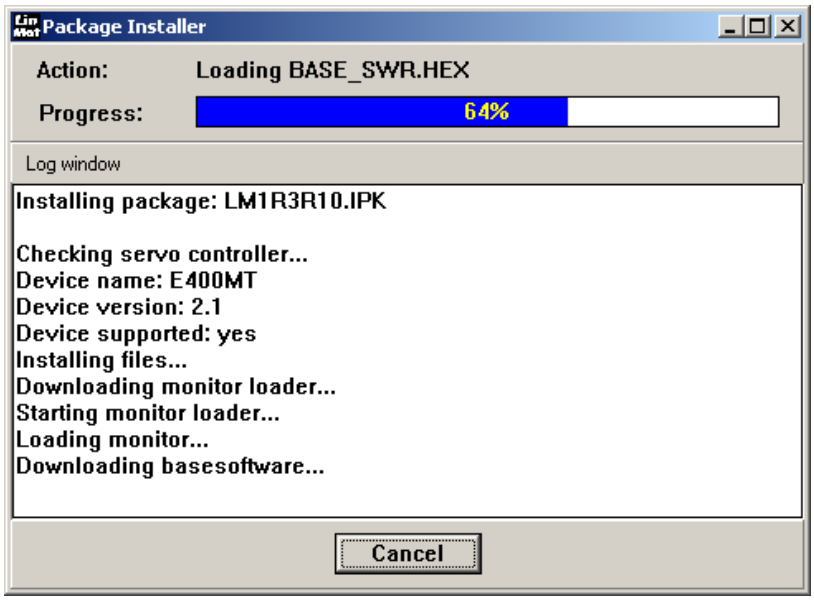


Figure 1-7: Package installer during the installation of *LinMot®* servo controller

1.7 I/O status display

The new I/O status display assists the user when commissioning the servo controller. It shows the status of the input and output signals. It enables verifying whether all inputs are connected properly. The display is activated by pressing the **IO Status** key in the **Error Inspector**.

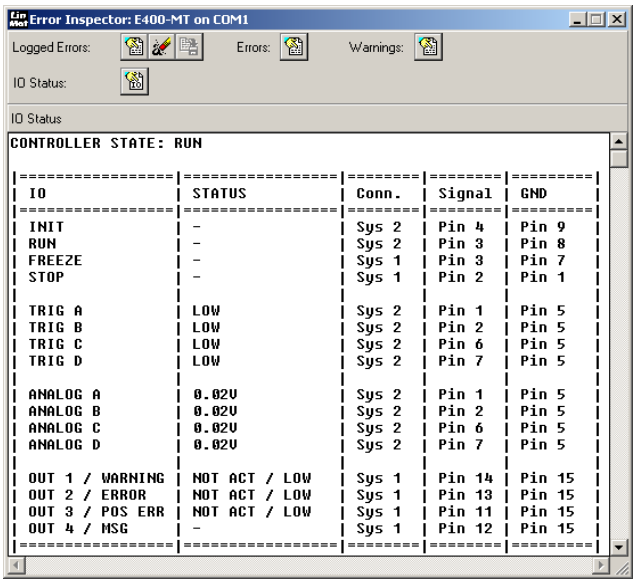


Figure 1-8: IO Status display

1.8 New commands for MT servo controller

Redefine Position — **Defining the actual position**

With this command the actual position is redefined. It may be performed only if the actuator in question has reached its target position and is not in the **FREEZE** state.

Set Current — **Set maximal current**

This command sets the maximum current and hence the force of the actuator concerned. Only positive values may be entered.

Set FF — **Setting feed-forward parameters (Release 1.3)**

This command sets the **feed-forward** parameters. It may be used to adapt the controller optimally when changing the load mass.

Set PID — **Setting PID parameters (Release 1.3)**

This command sets the PID parameters. It may be used to adapt the controller optimally during operation.

Set Cur. Offset — **Setting current offset (Release 1.3)**

This command sets the current offset for linear motors.

Set CP — **Setting curve properties (Release 1.3)**

This commands sets the motion profile properties offset, amplitude and speed.

Recipe

Change Recipe for Master Encoder (Release 1.3.10)

When operating in master encoder mode, this command changes the cams for the next cam cycle. This command is only supported in master encoder mode.

Start Cam

Change to cam mode (Release 1.3.10)

This command sets up the cam mode. It is possible to switch between time locked and position locked mode. This command is only supported in master encoder mode.

The table below provides an overview of the available MT commands:

MT commands	
Command	**Description**
No Operation	Do not obey any command
Abs. position	Positioning to absolute position
Rel. position	Move by position difference
Abs. current	Set absolute amperage
Rel. current	Set relative amperage
Set current	Set maximum amperage
Set FF	Set feed-forward controller parameters
Set PID	Set PID controller parameters
Set Cur. Offset	Set current offset
Set CP	Set motion profile properties
Curve	Run setpoint motion profile
Move home position	Move home position
Redefine position	Define actual position
Freeze / unfreeze	Interrupt movement
Stop	Stop movement
Recipe	Change recipe for cam applications
Start Cam	Change to cam mode

Abs Pos Act Pos

Goto absolute position from actual position. (since release 1.3.13)

Nearly the same command as Abs.Position but the velocity/acceleration limiter starts from the actual position. This command is intent to be used for releasing from a press situation.

SetDPosToAPos

Set demand position to actual position. (since release 1.3.13)

Sets the demand positions to the actual motor position. This command is used e.g. when the motor has been freezed and the motion should not continue when releasing from freeze if the motor has been current free and should be powered again without moving.

1.9 Operational states

The servo controller **DISABLE** state is now signalled by the two LEDs **Stat A** and **Stat B** flashing twice briefly. This is now distinguished clearly from the **STOP** state. The diagram below shows all conditions that must be satisfied for a change of state. Since release 1.3.9 the state **STOP** (emergency stop state) is distinguished by blink codes from the states **INIT** and **RUN**. The state **STOP** can only be left to the state **DISABLE** by clearing the INIT, RUN and STOP request flags..

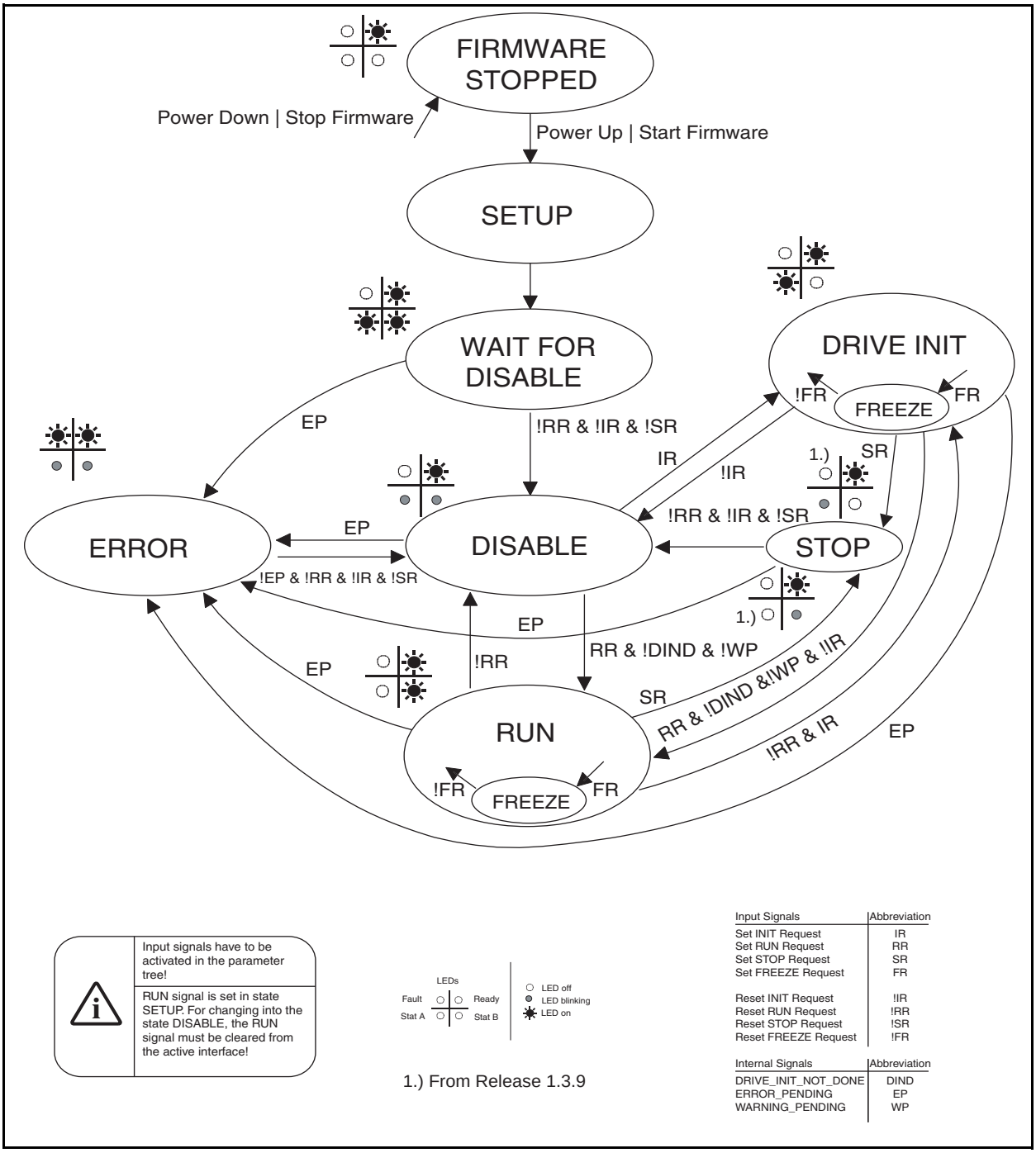


Figure 1-9: Operational states with LED display for version 2 controllers

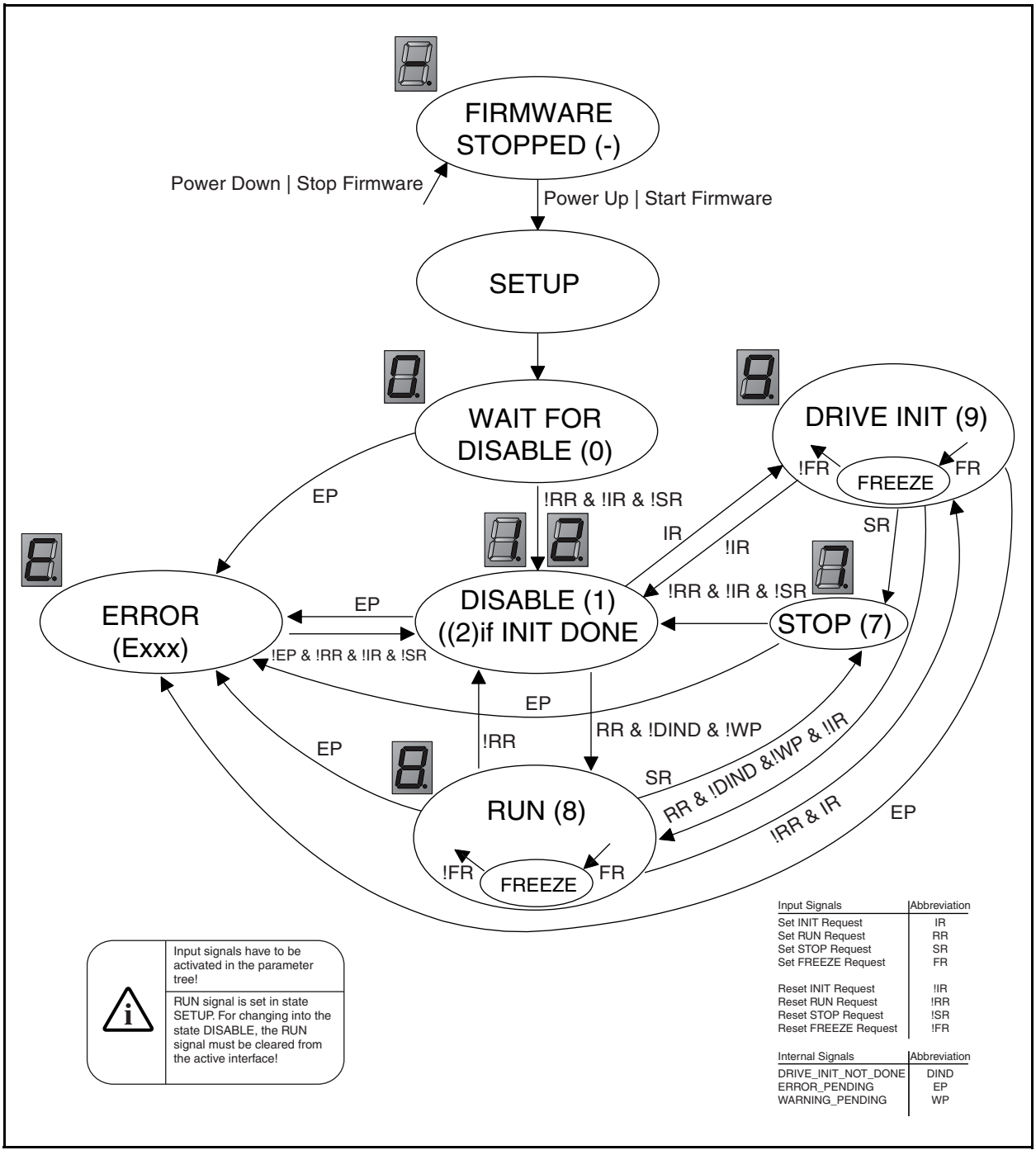


Figure 1-10: Operational states with 7 segment display for version 3 controllers

Powerup

The following diagram shows I/O signals during a typical powerup.

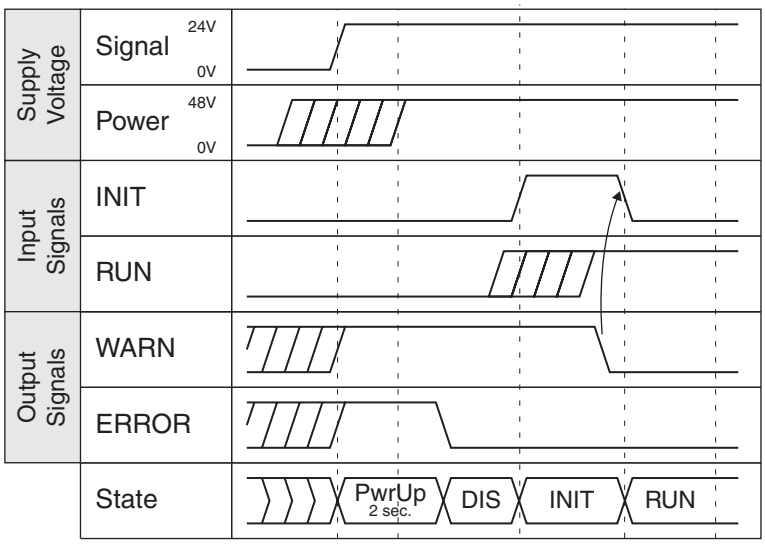


Figure 1-11: Powerup

Interruption of the power supply

Because of the safety regulations it is in most cases necessary to turn of the power supply of the motors in case of an emergency shutdown. The *LinMot*® units are well equipped for this procedure because they have separate motor power and logic signal supplies. Therefore in case of an emergency shutdown the power supply can be interrupted while the signal supply can be kept on so that the motors may not be initialized with homing. The following diagram shows the relevant I/O signals during such a procedure.

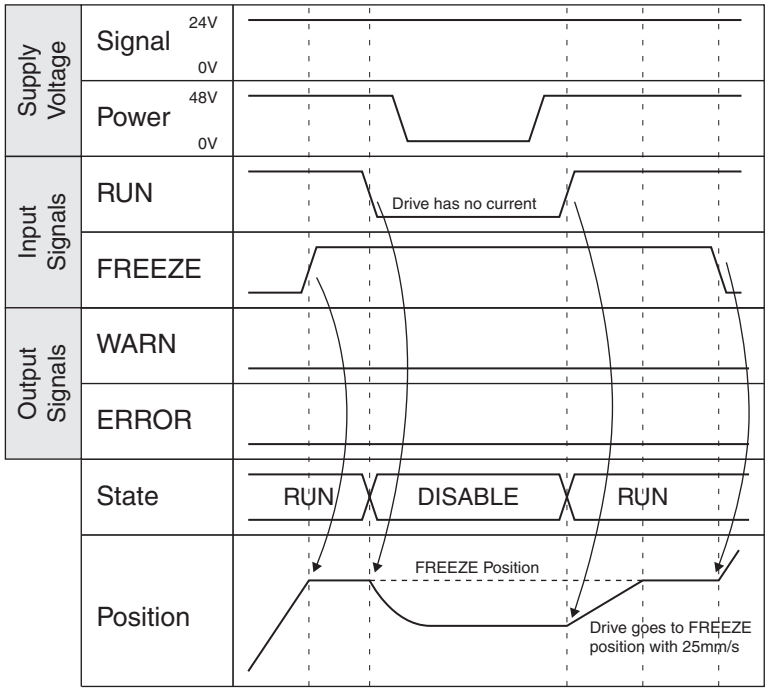


Figure 1-12: Interruption of the power supply

2. MT Servo Controller

2.1 Overview

The synchronization of electric drives is normally made by complex electronic encoders. Simple PLCs are flow controlled and don't provide this functionality. The multitrigger concept permits nevertheless the synchronization of complex movements using simple PLC systems.

Mechatronic Control Concepts

Electronic main shaft

Starting from an angle signal usually given by an angle encoder flanged onto the main shaft the electronic mainshaft causes all drives to follow in angular synchronism, i.e. position-controlled. Both central and decentral concepts are employed. In the latter case, each decentral drive receives an angle signal enabling it to read the required target position from a stored table. If flag errors are disregarded, the entire machine may be considered as rigidly intercoupled. The control outlay on machines engineered in this way is not to be underestimated, and calls for elaborate electronics. In particular a distinction must be drawn between the synchronous running of the drives required during operation which is disturbed only by lag errors, and the situation known as "special conditions" as occur when switching on the machine, during maintenance or faults.

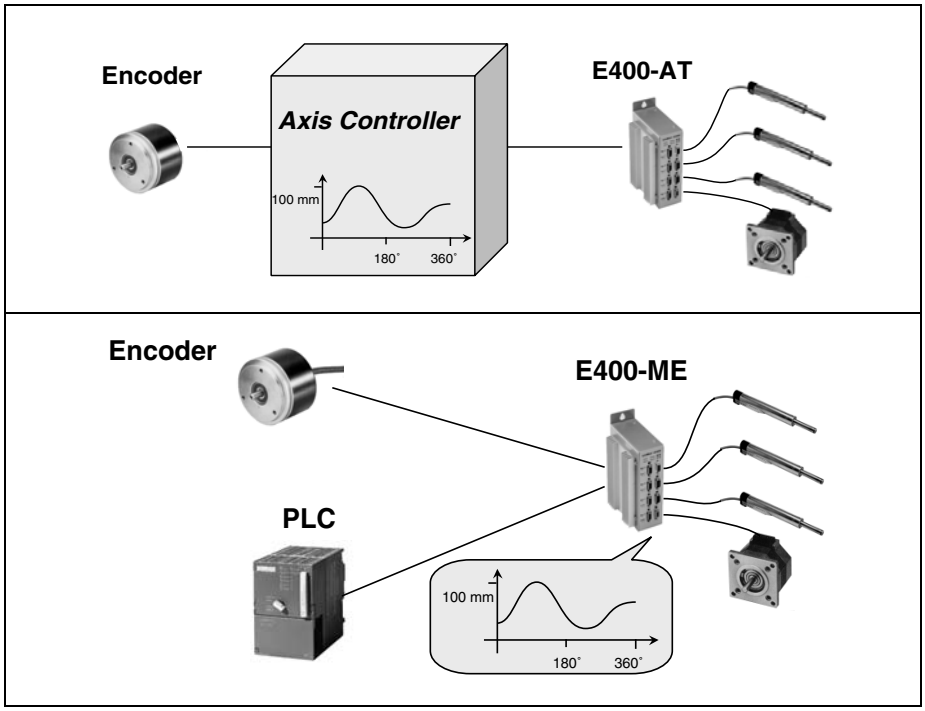


Figure 2-1: Electronic main shaft

Using electronic main shaft all drives are synchronized on the basis of an angle signal from an encoder. The motion profiles are stored centrally in an axis control or decentrally in the drives themselves.

Sequential control

A fundamentally different philosophy is followed by the sequential control, which may be compared with a cam control system. Here the individual functions are controlled not on the basis of angular information but by successive events or time intervals. This approach is very common in the world of PLC programming.

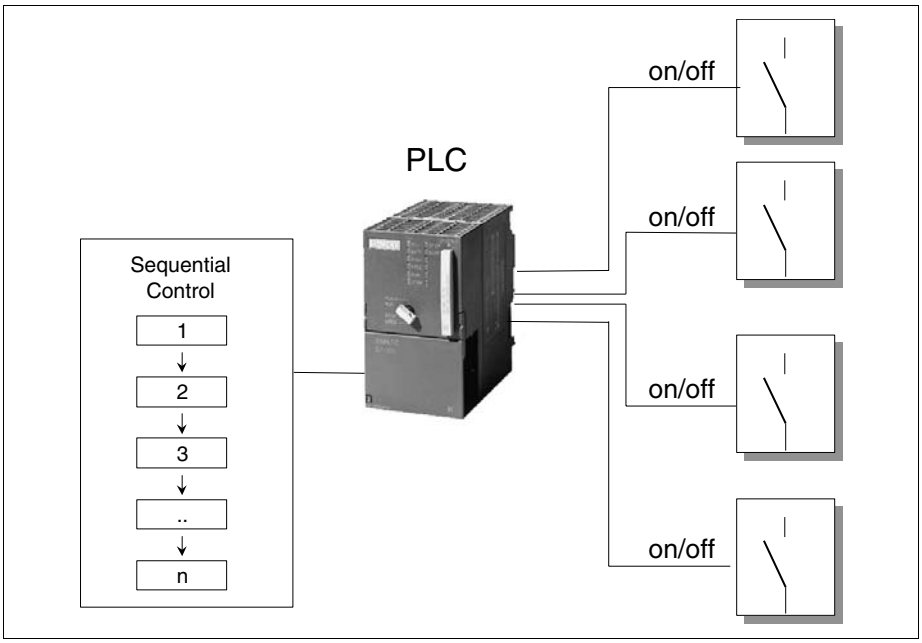


Figure 2-2: sequential control

Sequence controls run through the program step by step, the advance from one program point to the next being triggered by events or time intervals. Every program point is a self-contained operation, such as switching a relay on or off.

Compared with angle-synchronous control by means of electronic shaft, sequential control is much simpler, for all operational situations are handled by identical conception. This means there is no need to distinguish between angle-synchronous running in normal operation and the sequential procedures during initialization, maintenance or emergencies. On the other hand there are restrictions wherever processes, in special motions, must run parallel or synchronously with each other. A typical example are mechanical function units with several interacting motions for folding or assembling packagings and parts.

PLC combined with complex motion sequences

Today many control tasks in mechanical engineering are performed by simple PLC systems. Nevertheless once elaborate motions have to be controlled, the demands made on PLC systems are so exacting that the low-priced small devices are no longer up to the job. Mechatronic engineering has now reached the stage where the number of function modules realized by the closed loop brushless permanent magnet linear motors or rotational drives is growing steadily, so that increasingly complex motion sequences have to be controlled. In the view of this situation, the *LinMot®* servo controller has been extended to include a so-called multitrigger functionality. The basic idea behind multitrigger is to regard single or coupled motions of several motors as self-contained motion sequences. That means a movement or motion sequence of several coupled linear or rotary motors can be started and stopped by the PLC, comparable with a relay that is switched on and off.

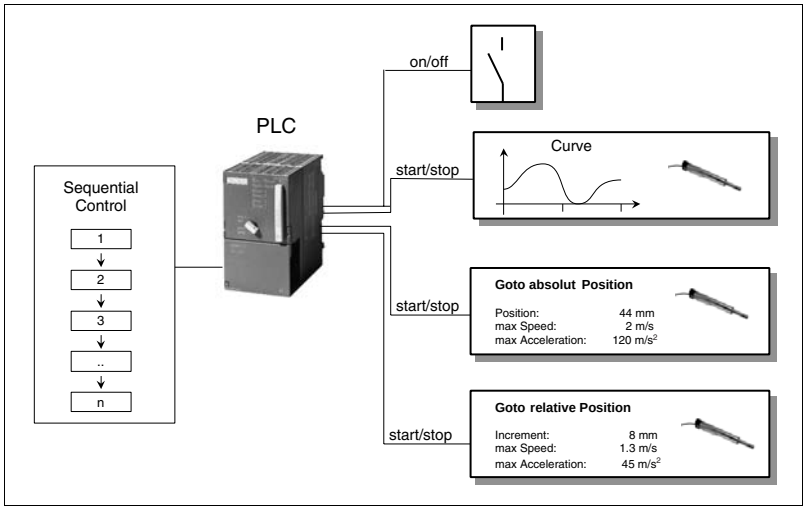


Figure 2-3: Sequential control

Sequential control runs through the program step by step, triggering complex motion sequences (absolute or relative movements, contours) comparable with the closing of a relay. These then proceed decentrally in the servo controller.

To confirm that the motion has been carried out, PLC is able to signal back like an end position switch or monitor a following error. In addition the PLC is capable of stopping a machine that has been started or interrupting it for a certain time. Otherwise the entire motion sequence is performed decentrally in the servo controller, so that the PLC is relieved entirely of the position controlling that involves much computing.

The figure above shows various motion sequences available from *LinMot®* Multitrigger with the associated configuration possibilities. Especially interesting with regard to combined movements is the ability to control up to 4 motors simultaneously with one E400-MT or E4000-MT servo controller. In this way the synchronous motion of 4 motors can be triggered by a single start command from the PLC. The following example will provide an insight into these possibilities.

Inserting a lid

Shown below schematically is a motion sequence as needed to insert a lid in a container:

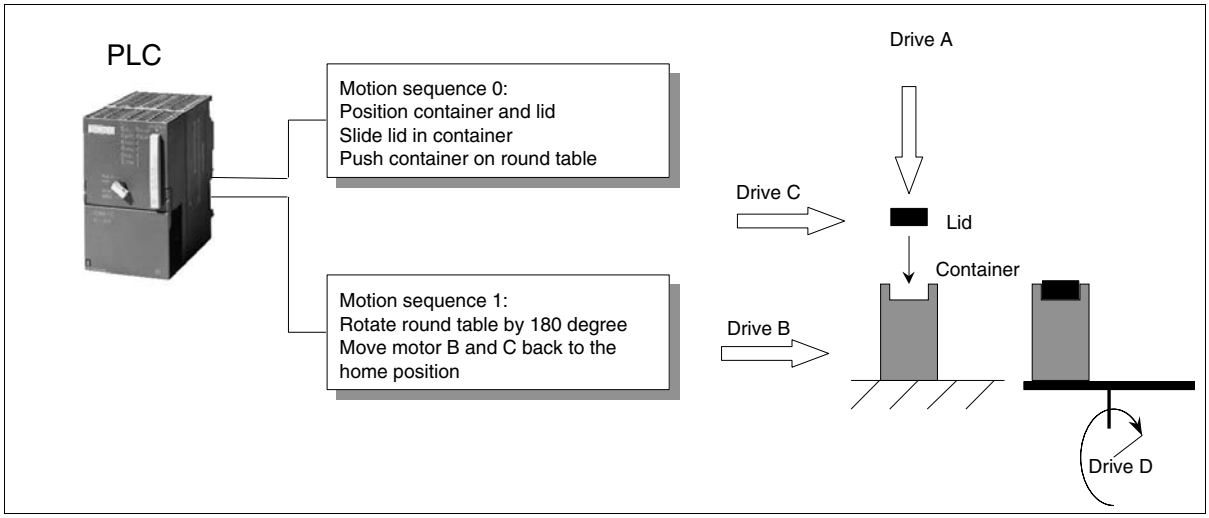


Figure 2-4: Inserting a lid

The two motion sequences are triggered by digital signals from the PLC which thus controls the entire sequence without having to bother about synchronization or the actual position control of the motors.

After a first start command from the PLC, the linear drives B and C bring the container and lid respectively into the working position. Linear motor A simultaneously begins a vertical motion profile which pushes the lid into the container. The container is then ejected onto a revolving table, and a corresponding feedback signal is sent to the PLC (motion sequence 0). Provided further conditions are fulfilled, the PLC starts the motion sequence 1 which returns drives B and C to their home positions and turns the table 18O° at the same time (drive D).

The next figure shows the programming of motion sequences 0 and 1 (states 0 and 1) with the *LinMot*® Multitrigger control and the motion profiles 1, 2 and 3 for motion sequence 0.

State	Drive A - LinMot	Drive B - LinMot	Drive C - LinMot	Drive D - Stepper
0	**Curve** Curve number 1	**Curve** Curve number 2	**Curve** Curve number 3	**No Operation**
1	**No Operation**	**Abs. Position** Position 0 mm Speed 1 m/s Acc. 50.068 m/s^2	**Abs. Position** Position 0 mm Speed 1 m/s Acc. 50.068 m/s^2	**Rel. Position** Increment 180 Steps Speed 800.018 Steps/s Acc. 10014.306 Steps/s^

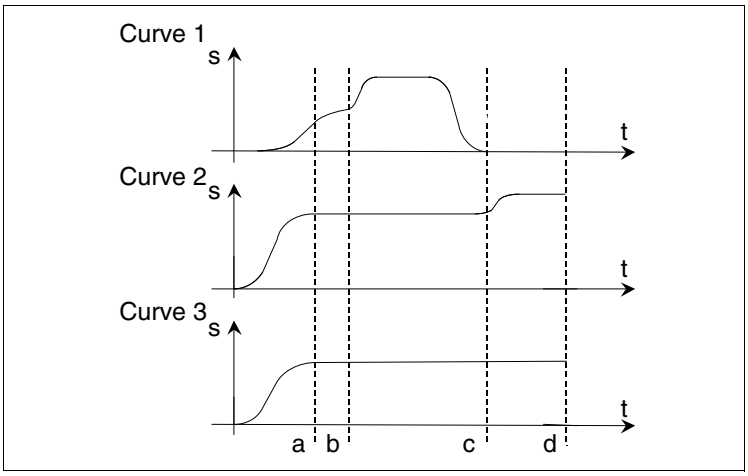


Figure 2-5: Programming the motion sequences with a *LinMot*® MT servo controller

Plotted on the right are the motion profiles of motion sequence 0 (state 1). These are to be understood as follows: motors B (motion profile 2) and C (motion profile 3) move the container and lid respectively to the working position (a). At the same time motor A (motion profile 1) moves vertically to the lid (b), presses this into the container and then withdraws to its home position (b to c). Motor B (motion profile 2) then ejects the container onto the indexing table (c to d). The motion sequences 'state 2' and 'state 3' define the positions of the motors in emergencies and during maintenance respectively.

Summary

Together with direct linear motors and stepper motors, the Multitrigger concept enables complex mechatronic function units to be achieved in simple fashion. The essential feature is the relieving to a large extent of the higher-level overall control system (PLC, PC), so that a simple and low-cost configuration is made possible

2.2 Setup and installation

In this section the control of the *LinMot*® MT servo controller by a master control system will be explained in detail. The Multi Trigger servo controller of the Ex00-MT and Ex000-MT series may be operated in the Multi Trigger or Analog Trigger mode.

Operation in Multi Trigger mode

The Multi Trigger mode is an extension of the Digital Trigger mode on the AT servo controller (see user handbook section 4.1.3). In the Digital Trigger mode, two setpoint motion profile can be followed by each motor using the rising and falling slopes of the trigger signal.

In the Multi Trigger mode up to 64 setpoint motion profiles or reference positions can be stored on the servo controller for each motor. They may be selected by means of digital trigger signals from the master control.

.

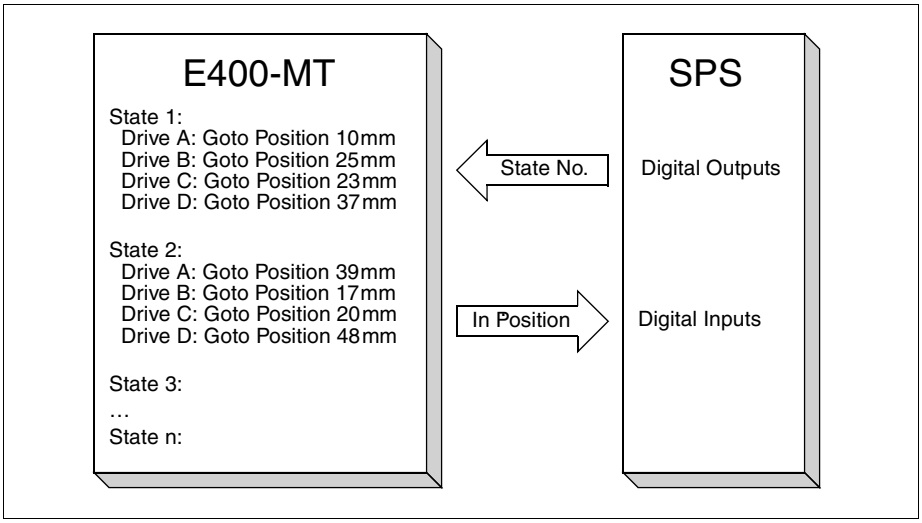


Figure 2-6: MT control by digital trigger signals

The movements are stored in the form of states (state 0, state 1.....state 63) on the servo controller. The master control (PLC) calls up the individual states by means of digital trigger signals. As soon as the actuators have reached the end positions of the current state, this is reported to the master control by means of digital **In Position** signals.

2.3 State Table

Up to 64 states may be defined in this table. Defined in each individual one of them are the movements that must be performed by the actuator concerned when the state is selected.

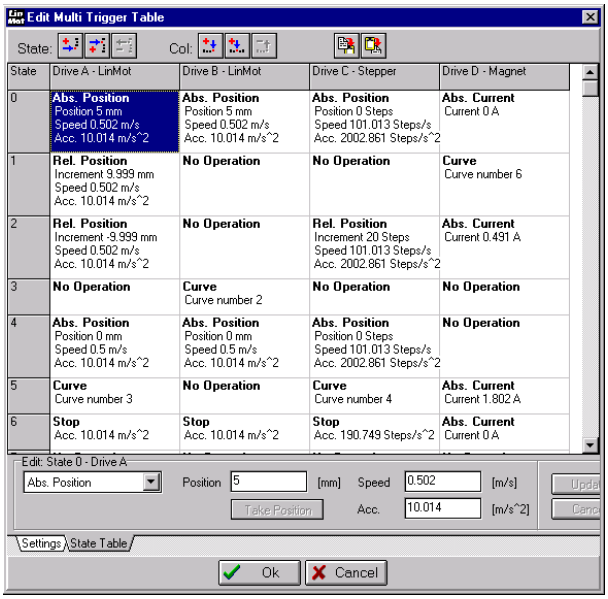


Figure 2-7: State Table

The movement to be performed can be defined in the table by the following functions:

No Operation — **Actuator performs no movement**

The actuator performs no movement or completes a movement already begun. This is used with servo controllers for several axes when an actuator is not to perform any movement in the particular state or is to complete the movement started.

Abs. Position — **Positioning to absolute position**

The actuator is brought to the desired absolute position (relative to zero) subject to an adjustable maximum speed and maximum acceleration. As soon as the actuator reaches its end position the **In Position** signal is activated.

Abs. Current — **Set an absolute amperage**

This command is visible only when controlling solenoids and serves to set the current for this.

Rel. Position — **Displacement about the relative position**

The actuator is displaced by the desired distance (relative to actual target position) not exceeding a preselected maximum speed and maximum acceleration. As soon as the actuator reaches its targets position the **In Position** signal is activated.

Rel. Current — **Setting a relative amperage**

This command is visible only when controlling solenoids and serves to set a relative current for them.

Curve

Run motion profile

A stored motion profile on the servo controller is run subject to an adjustable maximum speed and maximum acceleration. As soon as the actuator reaches the last setpoint on the motion profile, the **In Position** signal is activated.

Move Home Position

Displacement of the reference position (zero)

The motion profiles and absolute positions relate always to the reference position determined after initialization. With **Move Home Position** the reference position (zero) of the drive in question is displaced by the desired distance. This command may be executed only if all actuators have reached their setpoints and none of the motors are in the **FREEZE** state.

Redefine Position

Defining the actual position

With this command the actual position is redefined. This command may be executed only if the actuator concerned has reached its target position and is not in the **FREEZE** state.

Stop

Stop movement

The movement in progress is interrupted and the actuator is brought to a stop subject to an adjustable maximum acceleration. As soon as the actuator is stopped, the **In Position** signal is activated.

Freeze/Unfreeze

Movement interruption

The freeze command interrupts the movement in progress, and the actuator is brought to a stop subject to the maximum acceleration for the current movement. Unlike the stop command the **In Position** is not activated by the freeze command.

The unfreeze command enables the interrupted movement to be completed. Once this has been done, the **In Position** signal is activated. If commands are called while the actuator is in the freeze state, upon giving the unfreeze command the command last called is executed (if the commands last called were **Rel. Position** commands, the relative positions are added).

All **FREEZE** commands are cancelled upon leaving the **RUN** operating state.

Set Current

Setting maximum current

This command sets the maximum amperage and with it the force of the actuator in question. Only positive values may be set.

Set Cur. Offset

Setting current offset

This command sets the current offset. It can be used to compensate a static force.

Set FF

Setting the Feed Forward Parameters

This command sets the **Feed Forward** parameters. It can be used to obtain optimal adaptation of the controller when changing the load mass.

Set PID

Setting the PID parameters

This command sets the PID Parameters. It can be used to adapt the controller to changing load conditions during operation.

Set CP

Setting the Motion Profile Properties

This command sets the motion profile properties offset, amplitude and speed.

Recipe

Change Recipe for Master Encoder (since release 1.3.10)

When operating in master encoder mode, this command changes the cams for the next cam cycle. This command is only supported in master encoder mode.

Start Cam

Change to cam mode (since release 1.3.10)

This command sets up the cam mode. It is possible to switch between time locked and position locked mode. This command is only supported in master encoder mode.

Start Cam ActPos

Change to cam mode and start cam from actual position (since release 1.3.11)

This command sets up the cam mode. It is possible to switch between time locked and position locked mode. This command is only supported in master encoder mode. In contrast to the Start Cam command this will set the curve position offset parameter such as the cam start point is equal to the actual wanted position.

Curve ActPos

Run motion profile from actual position (since release 1.3.11)

Same as Curve command but the curve position offset parameter will be set such as the curve starts at the actual wanted position.

Abs Pos Act Pos

Goto absolute position from actual position. (since release 1.3.14)

Nearly the same command as Abs.Position but the velocity/acceleration limiter starts from the actual position. This command is intent to be used for releasing from a press situation.

SetDPosToAPos

Set demand position to actual position. (since release 1.3.14)

Sets the demand positions to the actual motor position. This command is used e.g. when the motor has been freezed and the motion should not continue when releasing from freeze if the motor has been current free and should be powered again without moving.

2.4 Settings Table

The individual states are called up by the master control (e.g. PLC) with four digital signals (**Trig In 1…4**). A command is assigned to each of the 16 possible input combinations of the trigger signals. These commands allow states to be controlled directly, calling up the following state or the previous one, or performing the same state once again.

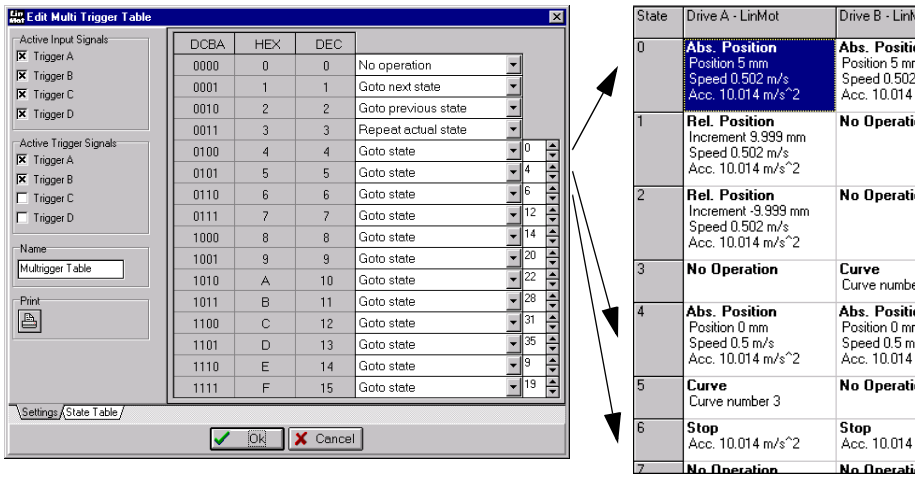


Figure 2-8: Controlling individual states

No operation

If the input signals change to an input combination for which the **No operation** command is defined, the current state is retained.

If the movement of the current state is not completed yet, the movement commenced is ended as defined in the current state.

Goto next state

If the input signals change to an input combination for which the **Goto next state** command is defined, the state following the current one is executed.

In table 2-1, “Resulting movements,” on page 23 it will be seen what movements result when **Goto next state** is called during a movement in progress (of the previous state).

Goto previous state

If the input signals change to an input combination for which the **Goto previous state** command is defined, the state preceding the current one is performed.

If the movement of the current state is not yet completed, the resulting movements may be obtained from the table above.

Movement started	Movement called	Resulting movement
Abs. Position A	No Operation	Abs. Position A
	Abs. Position B	Abs. Position B
	Rel. Position +B	Rel. Position A+B
	Curve 2	Curve 2

Table 2-1: Resulting movements

Movement started	Movement called	Resulting movement
Rel. Position +A	No Operation	Rel. Position +A
	Abs. Position B	Abs. Position B
	Rel. Position +B	Rel. Position A+B
	Curve 2	Curve 2
Curve 1	No Operation	Curve 1
	Abs. Position B	Abs. Position B
	Rel. Position +B	last demand value of Curve 1 + B
	Curve 2	Curve 2
Stop	No Operation	Stop
	Abs. Position B	Abs. Position B
	Rel. Position +B	Rel. Position +B
	Curve 2	Curve 2
Freeze	No Operation	Freeze
	Abs. Position B	Freeze
	Rel. Position +B	Freeze
	Curve 2	Freeze

Table 2-1: Resulting movements

Repeat actual state

If the input signals change to an input combination for which the **Repeat actual state** is defined, the current state is performed once more. If this state is to be repeated a number of times, between the repeat actual state commands the input combination for the **No operation** command must be given each time, so that a change of the input combination ensues and hence **Repeat actual state** is called.

If the movement of the current state is not yet completed, the resulting movements may be obtained from the table above.

Goto state

If the input signals change to an input combination for which the **Goto state** is defined, this state is carried out.

If the movement of the current state is not yet completed, the resulting movements may be obtained from the table above.

From the example set out in the table below it will be clear how the desired states are called up from the master control by means of the four digital signals **Trig In 1…4**. A new state is performed only after the input combination of the trigger signals has been stable for an adjustable time (jitter filter).

Shown in the table are the movements of motor A from the foregoing chart,

TRIG IN 4/3/2/1 input signals	Command	Current state number	Movement of motor A
0100	Goto State 0	0	Positioning to absolute position 5mm • max traversing speed 0.5m/s • max acceleration 10m/s^2
0001	Goto next state	1	Positioning to absolute position 10mm • max traversing speed 0.5m/s • max acceleration 10m/s^2
0000	No Operation	1	-
0001	Goto next state	2	Positioning to relative position -10mm • max traversing speed 0.5m/s • max acceleration 10m/s^2
0000	No Operation	2	-
0011	Repeat actual state	2	Positioning to relative position -10mm • max traversing speed 0.5m/s • max acceleration 10m/s^2
0000	No Operation	2	-
0001	Goto next state	3	Slider remains in current position or commenced movement is completed
0000	No Operation	3	-
0110	Goto State 6	6	Slider is stopped with acceleration 10m/s^2
…	…	…	…

Table 2-2: Sequence of individual states

Operation in Analog Trigger mode

In the Analog Trigger mode the MT servo controllers Ex00-MT and EX000-MT behave like the AT servo controllers Ex00-AT and Ex000-AT. With this setting the MT functions described in these instructions are not available - only the AT functions explained in the user handbook.

2.5 Configuration software

The Multi Trigger servo controllers Ex00-MT and Ex000-MT are configured with *LinMot*® Talk configuration software like the Analog Trigger servo controllers Ex00-AT and Ex000-AT. For programming the MT servo controller the **Curve Inspector** has been extended with a graphical user interface for the Multi Trigger functions. All the functions of the *LinMot*® Talk described in the operating instructions remain the same, and will not be enlarged upon further here.

Curve Inspector

This section will deal only with the extension of the **Curve Inspector** for the Multi Trigger table. All other functions of the **Curve Inspector** may be read up in the operator's handbook.

Besides the curves, with the **Curve Inspector** Multi Trigger tables may be defined also for the MT servo controller. In the **Curve Inspector** these are handled like normal curves, can be edited like curves, loaded onto the servo controller and run. If motion profiles are called in the Multi Trigger table, they must be loaded onto the servo controller together with the Multi Trigger table. Unlike the motion profiles, only one Multi Trigger table may be loaded onto a servo controller.

Multi Trigger Table

If the MT servo controller is configured for the Multi Trigger mode, in the **Curve Inspector** there is a **Create Multi Trigger Table** key available beside the **Create Curve** key.

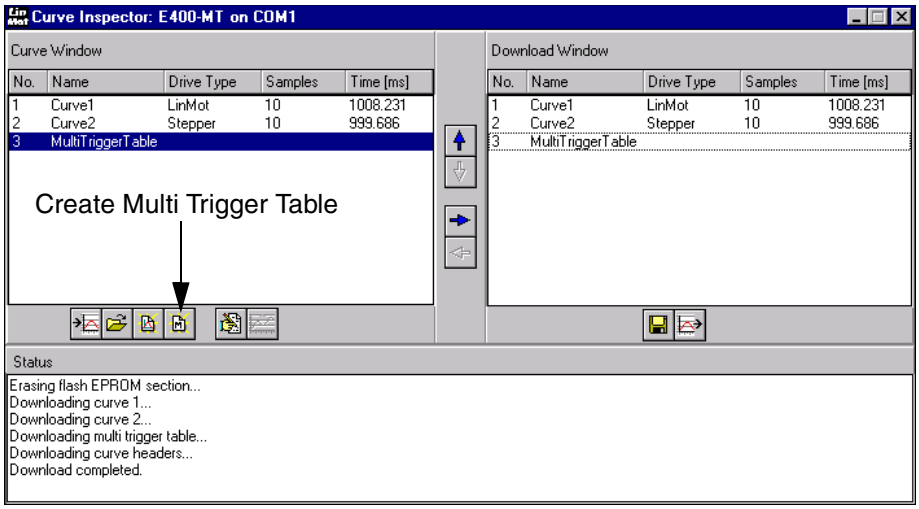


Figure 2-9: 'Create Multi Trigger Table' key

Pressing the **Create Multi Trigger Table** key opens the **Edit Multi Trigger** window.

The **Edit Multi Trigger** window consists of two pages, designated with the two tabs **Settings** and **State Table**.

In the **State Table** the reference positions and setpoint motion profiles are entered as well as the desired commands for the individual motors. The digital input signals and input combinations are assigned to the individual states on the settings page.

Control elements in 'Settings'

The first page of the **Edit Multi Trigger** table serves to set the active triggers and to define the name of the Multi Trigger table and the input table.

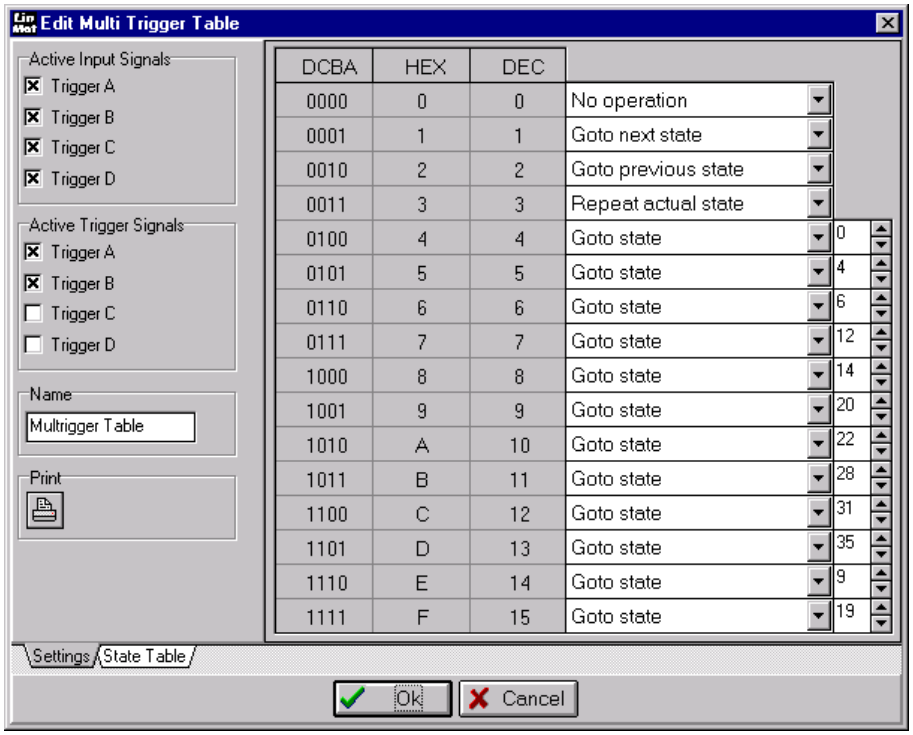
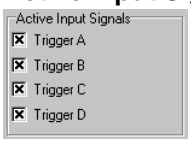


Figure 2-10: Settings

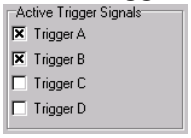
Active Input Signals



The **Active Input Signals** define which input signals are controlled by the master control system. If all four input signals are activated, sixteen input combinations are possible (0-15) for which an appropriate command must be defined in the command table.

Important: The activated trigger signals are active only in the **RUN** operating state. If the trigger inputs are needed for initializing, they must be activated in the parameter inspector under **System\IO Configuration**.

Active Trigger Signals



Through the fields of the **Active Trigger Signals** the input signals are selected whose change of level will cause the appropriate command to be executed. With the adjustment in the right of the illustration, the levels of input signals C and D may change without a new command being executed. Only with a change of input signal A or B respectively will the command defined for the actual input combination be executed.

Name

A name may be assigned to the **Multi Trigger** table. The desired name is entered in the field.

Print

Upon pressing this button the entire configuration of the **Multi Trigger** table is printed out.

Command table

In this table, commands are assigned to all combinations of the input signals. This allows for example a state to be called directly in the table of state (see next section), in the state following or in the previous one, or the current state to be repeated.

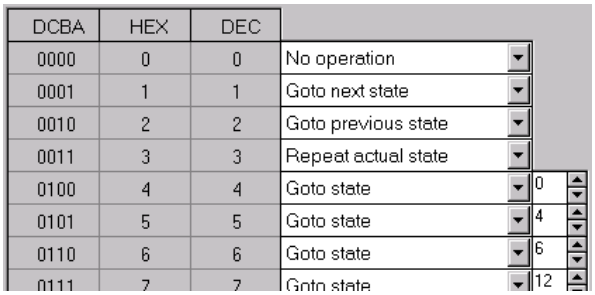
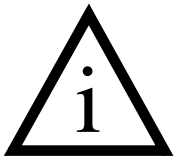


Figure 2-11: Command table

Entered in the first three columns are the input signal levels for all input combinations in binary, hexadecimal and decimal form. Via the pull-down menu in the fourth column, the desired commands may be set for the input combinations. The fifth column appears only if a state is called directly with the **Goto state** command. The desired state number (0-63) can then be set in the fifth column.



In the fifth column state numbers may be set only if the states in question have already been defined in the table of states.

In the table of commands, commands may be assigned to the input combinations only if all necessary input signals have been activated. Input combinations not valid because the corresponding input signals have not been activated are shown in grey in the command table.

Control elements in “State Table”

Defined in the **State Table** are the actions to be performed by each individual actuator in the particular status. Up to 64 states may be stored. At any state one or all motors can be actuated.

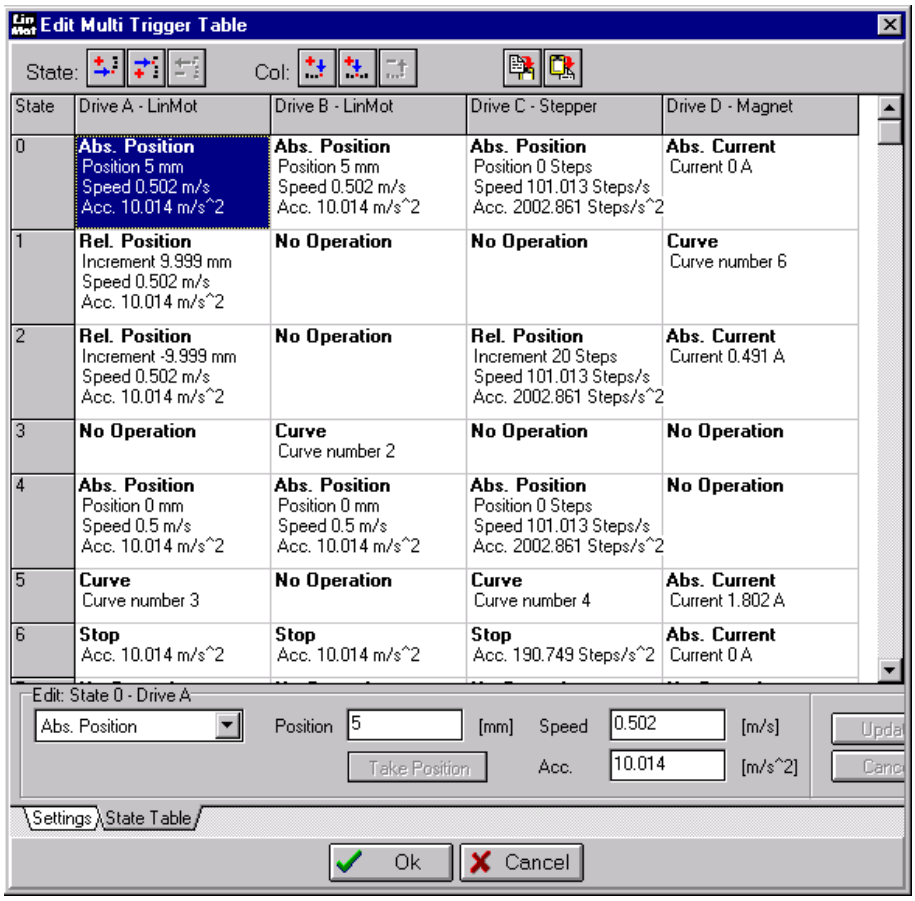


Figure 2-12: Table of states

Add State

A new state may be added at the bottom of the table. Adding a new state is possible only if the table of states has less than 64 lines.

Insert State

A new state is inserted at the top of the one selected momentarily. A new state may be inserted only if the table of states has less than 64 lines.

Delete State

The selected state is deleted. This icon is active only if the entire line of the state in question is selected. More than one line may be deleted at a time. Lines once deleted cannot be recalled. To prevent unintentional deletion the deleting of lines must be confirmed.

Add Column

A new column for a further actuator is added at the right-hand side of the table. More actuators may be added than the connected servo controller is capable of controlling. This enables a drive to be defined in a new column and copied later into another column.

Insert Column

A new column for a further actuator is inserted at the left of the selected column. More actuators may be inserted than the connected servo controller is able to control. This enables a drive to be defined in a new column and copied later into another one.

Delete Column

The selected column is deleted. This key is active only if the entire column is selected. More than one column may be deleted at a time. Columns once deleted cannot be recovered. To prevent unintentional loss, deletion must be confirmed.

Copy

The entries of individual fields, several fields, entire lines or columns may be copied.

Paste

By means of the **Paste** command the entries of the copied fields may be inserted in other fields. The **Paste** command functions only if the actuator types of the copied field and insertion field and the number of fields are identical.

State commands

In the individual fields of the state table the movements or commands are entered which the actuator in question is to perform when the state is called.

Figure 2-13: Entering state commands

The state command is selected in the selected field of the table on the left-hand side by means of a pull-down menu containing all state commands valid for the motor type adjusted.

The right-hand fields beside the pull-down serve to configure the state command. Only the fields available with the state command selected are displayed.

By means of the **Update** key the adjustments made are transferred into the field. If they are not to be taken over, the alterations can be rejected by pressing the **Cancel** key.

Store/Close

With the **Ok** key the **Edit Multi Trigger** window is closed and the alterations are stored in the Multi Trigger table. The **Cancel** key enables the window to be closed without storing the alterations.

After closing the **Edit Multi Trigger** window the Multi Trigger table is in the **Curve Window** of the **Curve Inspector**. The **Multi Trigger** table can now be moved like a motion profile into the download window and loaded onto the servo controller.

Set Port and Drive Type

After a double click at the first cell of a column, the **Set Port and Drive Type** window appears.

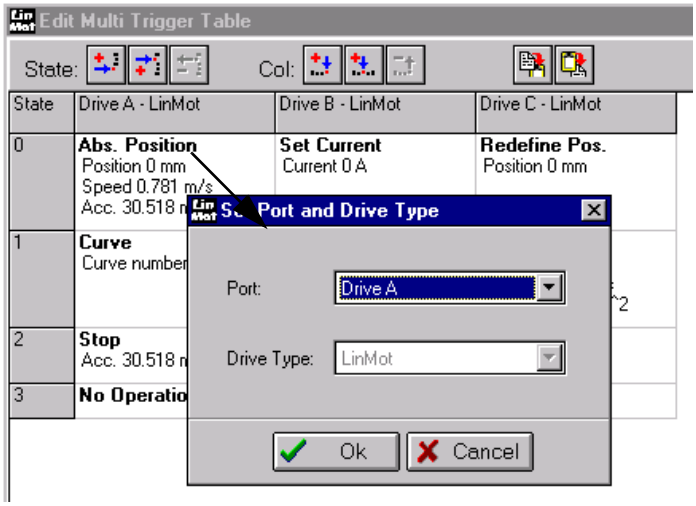


Figure 2-14: “Set Port and Drive Type”- Window

In this window the user may alter the port (motor output on servo controller) for the selected column and actuator type. However, the actuator type can be altered only if no cell of the column has been edited yet. By pressing the **Ok** key the alterations are stored, with the **cancel** key they are cleared.

Columns may also be defined that are not assigned to any motor output. This allows columns to be exchanged in the state table by altering the columns in question to **Not Assigned** first and then assigning them to the desired motor outputs.

Teach-In with Multi Trigger mode

The Multi Trigger servo controller of the Ex00-MT and Ex000-MT series may be programmed by teaching-in. This procedure allows very fast and precise programming of movement sequences. When teaching-in, the slider is brought manually to the desired reference position and the actual position is read out from the servo controller and entered into the desired field in the table of states.

The teach-in function is available in the absolute position mode in the state table. The actual slider position is taken over by the **Take Position** key. It must then be defined at what speed and with what acceleration the stored position is to be approached. By pressing the **Update** key the values are entered in the cell of the state table.

Figure 2-15: “Take Position” key for teach-in

For taking over the actual position values the connected MT servo controller must be started. After initializing the motors are switched off (by deactivating the **RUN** input). Now the sliders can be brought to the desired reference positions and the values transferred into the table of states.

3. PROFIBUS Servo Controller

Release 1.3 now supports the PROFIBUS-DP-compatible *LinMot®* servo controller. These servo controllers have a 12MBit/s fast PROFIBUS-DP interface and are eminently suited for complex motion sequences in conjunction with PLC controls having an integrated PROFIBUS-DP master interface.

The PROFIBUS description is divided into the following subsections:

- chapter 3.1 “Overview”
- chapter 3.2 “State machine”
- chapter 3.3 “Cabling”
- chapter 3.4 “Commissioning without PROFIBUS”
- chapter 3.6 “Setup”
- chapter 3.7 “Data module overview”
- chapter 3.8 “Data modules”
- chapter 3.9 “Diagnosis”
- chapter 3.10 “Trouble shooting / remedying faults”
- chapter 3.11 “Interfaces”

3.1 Overview

Open field bus

PROFIBUS is an open field bus standard (EN 50170) that is finding ever more widespread use in automation. There are three versions of it: FMS, DP and PA. PROFIBUS-DP has been specially optimized for speed and is therefore especially suited for the higher-level control of high-dynamic motors like *LinMot®*.

Master-Slave

PROFIBUS-DP functions on the master-slave principle with overlaid token passing between different masters. The master-slave communication is strictly cyclic, whereby with the aid of time monitoring of the bus the failure of either a master or a slave is detected at once. In addition the diagnose of a slave by a master is standardized and offers considerable possibilities for transmitting error and state information.

A PROFIBUS-DP participator is identified via an adjustable address (0-125). Furthermore each equipment class has a so-called ident number, which is identical for all equipment of the same type (all *LinMot®* DP servo controllers have the same ident number).

www.profibus.com

In the description that follows it is assumed that the user possesses basic knowledge on PROFIBUS-DP. One very good information source for PROFIBUS information is the WWW address **http://www.profibus.com.** To be found there are various descriptions and further-reaching literature references.

3.2 State machine

To make the documentation understood more easily, a brief description of the state machine of a DP slave is given below. It shows the states in which a DP slave may be, and which steps it must pass through to go on line. On-line means the state in which the master exchanges useful data cyclically with the slave.

What is a Slave?

According to PROFIBUS terminology the *LinMot*® servo controllers are slaves. Therefore they cannot initiate data transmissions on their own but must be requested to do so by the so-called master, usually a PLC.

What is a Class 1 master?

A master of Class 1 conducts useful data traffic with the slave assigned to it. Class 1 masters are usually industrial PLC systems.

What is a Class 2 master?

A master of class 2 is intended for commissioning purposes and may briefly take over the control of any slaves. Class 2 masters are usually PCs with a PROFIBUS printed circuit board.

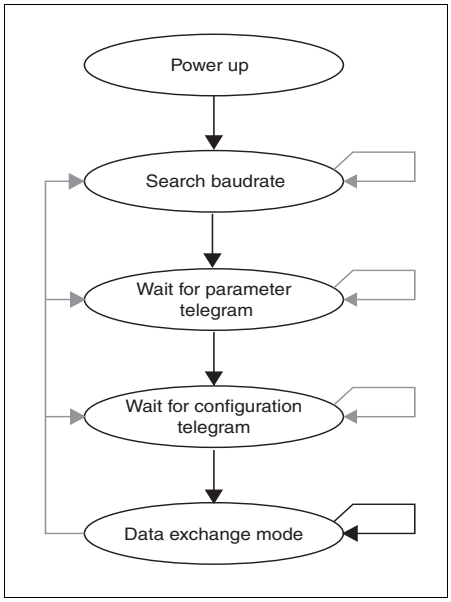


Figure 3-1: State machine of a PROFIBUS-DP slave

State	Description
Search baud rate	In this state the slave seeks the baud rate at which communication takes place on the bus.
Await parametring telegram	In this state only parametring telegrams are accepted by the slave. This telegram contains the information laid down in the standard, e.g. PNO number, sync-freeze capability etc. *LinMot*® servo controllers await no application-specific parameter data.
Await configuration telegram	The configuration telegram lays down the number and nature of the input and output data. *LinMot*® servo controllers support various data modules which may be put together at will. It can thus be decided when parametring which data shall ultimately be transmitted in the data exchange mode. For example a motor may be configured so that the target and actual positions are transmitted. Another motor on the same servo controller may be configured so that the maximum speed as well as the target position is transmitted.
Data exchange	When both the parametring and the configuration have been accepted from the slave's firmware, the slave assumes this state and exchanges useful data cyclically with the master.

3.3 Cabling

In this subsection hints and rules are given for correct cabling of the PROFIBUS network.

Shielding

Only cables with braided shielding should be used. The shielding must be large-area on both sides. With permanently installed equipment it is a good thing to bare the shielded cable without interruption and lay it on the shielding rail or earthed conductor rail. This will enhance reliability in an environment subject to severe interference.

Bus connector

Only bus connectors suitable for PROFIBUS and the corresponding baud rate should be used. The connectors at both ends of the bus should have connectable termination. Such connectors are obtainable from Siemens for example.

Potential equalization

The shielding of the PROFIBUS cable must not be used for potential equalization. For installations earthed at different points a separate earth line must be laid having an impedance at least 10 times less than that of the cable shielding.

Termination

With baud rates above 1.5 MBit/s the PROFIBUS must be terminated actively at both ends. In addition there should be a 100nH series inductance in each connector for each outgoing data line.

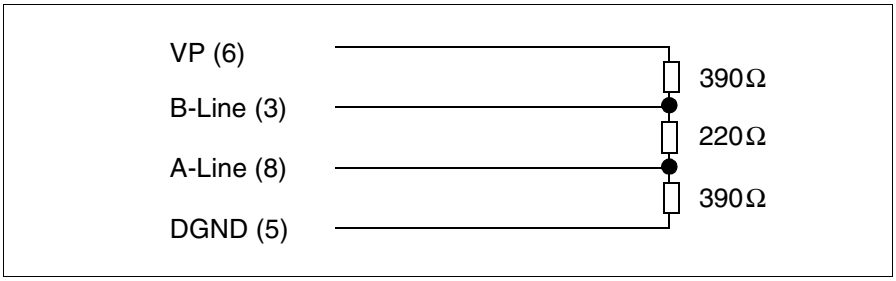


Figure 3-2: Termination of PROFIBUS line in accordance with EN50170 (Pin No. with 9-pin D-SUB connector)

Connector allocation

The connector allocation is laid down in the PROFIBUS standard. All equipment conforming to the standard must adhere to this allocation. The table below shows the allocation of the 9-pin DSBUN connector.

Pin No.	Signal	Significance
1	Shielding	shielding / safety earth
2	M24	earth of 24V safety earth
3	RxD/TxD-P[1]	receive / send data -plus, B-line, red
4	CNTR-P	repeater control signal (direction control)
5	DGND[1]	data transmission potential (reference potential to VP)
6	VP[1]	supply voltage-plus, (P5V)
7	P24	plus 24V output voltage
8	RxD/TxD-N[1]	receive / send data-N, A-line, green
9	CNTR-N	repeater control signal (direction control)

1) This signal must be present. The others are optional.

With complicated and widely distributed arrangement of PROFIBUS equipment on a bus line it is advisable to thoroughly study the technical guideline for building-up PROFIBUS-DP/FMS networks. This handbook may be ordered from all PROFIBUS user organizations under No. 2.111. The addresses of PROFIBUS user organizations are given on Internet under **http://www.profibus.com.**

3.4 Commissioning without PROFIBUS

The PROFIBUS servo controller can be commissioned without a working PROFIBUS connection. This mode is well suited for adjusting the controller and testing the cabling and the power supply. In this commissioning mode a loaded motion profile is running continuously.

The following steps explain how to proceed:

1. Set PROFIBUS address to 'EE'. Because the address switches are read only once at start up, it is necessary to reset the controller when address has changed.
   WARNING: If the address is set to "FF" the servo controller is in the so called bootstrap mode and no LED will be on (even not the green one).
2. Create motion profile with the *LinMot®* Talk software and download it to the servo controller. The start point of the profile should be equal to the last point.
3. Set motor type, initialization and controller parameters. See also chapter "Parameters" on page 60.
4. Set parameters in the directory **\Drives\Drive X\Set Value Generation \Set Value Configuration**. If the parameter **Curve Number** is set to 0 no motion profile will be executed!
5. In the **Control Panel** press first the **Stop** key and then the **Start** key.

If the cabling and all the parameters are correct the motor will now run the chosen motion profile cyclic.

3.5 PROFIBUS Parameters

The PROFIBUS relevant parameters are described in chapter 6.9.

3.6 Setup

PROFIBUS slave projects are performed mostly using PC setup software. All leading PLC makers, such as Allen Bradley, Bosch, Mitsubishi, Omron and Siemens provide such project environments.

Device Data Base Sheet

The basis for open setup is provided by the electronic data pages designated as equipment master files in the PROFIBUS standard. All information needed for setup the particular slave is taken over by the setup software from these equipment master files. The definition of the GSD files laid down in the standard ensures that every norm conforming slave can work with all norm conforming masters. The GSD file for the *LinMot®* servo controllers is named **LINM00B6.GSD** and is located according to the software installation in folder …/GSD of the installed *LinMot®* Talk software (e.g. C:/LINMOT/1R3/GSD/…).

Setup

The actual setup is done mostly by means of **drag and drop**. The following steps are run through typically:

1. Load all necessary GSD files of the slaves employed into the setup software. This step must be performed only once with the usual programs.
2. Create and configure a PROFIBUS system on the desired master.
3. Create the individual slaves on the bus system.
4. Configure the individual slaves.

When configuring the slaves the desired data modules may be determined which are exchanged with the master in cyclic traffic. All data modules supported by the *LinMot*® servo controller are described on page 39.

In Figure 3-3, “LinMot® PROFIBUS servo controller setup” a typical setup of a PROFIBUS system with two *LinMot*® servo controllers is shown. The project shown was accomplished with the help of the Siemens Step7 software.

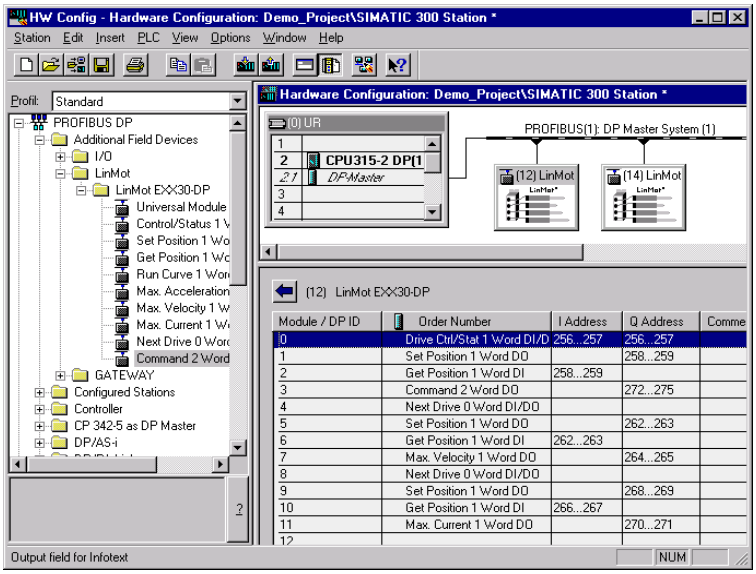


Figure 3-3: *LinMot*® PROFIBUS servo controller setup

3.7 Data module overview

This subsection provides an overview of the data modules that can be used to configure *LinMot®* PROFIBUS servo controller. With the help of these modules it can be defined which data are to be exchanged between the PROFIBUS master and the *LinMot®* servo controller. One module encapsulates an indivisible data block. Such a block may comprise one or more data values. The table below gives an overview of the available modules.

Data module	Description
Command	Performs a command on the *LinMot®* servo controller.
Control/Status	Transmits the control and status words to and from the servo controller respectively.
Get Position	Reads the actual motor position.
Get Current	Reads the actual current.
Max. Acceleration	Sets the maximum acceleration.
Max. Current	Sets the maximum amperage/power.
Max. Speed	Sets the maximum speed.
Next Drive	Introduces a new motor.
Run Curve	Starts a filed motion profile.
Run Incr. Curve	Starts a filed motion profile and the actual demand position is set as curve offset.
Set Position	Sets motor's target position.
Set Curve Speed	Sets speed of motion profiles.
Set Curve Amplitude	Sets the amplitude of motion profiles.
Set Curve Offset	Sets the position offset of motion profiles.

In the projecting software the modules bear after their actual name an identification indicating the direction of communication and the amount of data exchanged.

3.8 Data modules

All data modules are explained in the description below.

Command

This module serves to send commands to the *LinMot*® servo controller. It may be setup only once per servo controller. Since it is possible to specify at the running time for which motors the command is meant, this is only a minor limitation.

Module: Command	
Command ID	
Direction	Master ➟Slave
Size	1 word
Structure	

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Motor D	Motor C	Motor B	Motor A	Start Command (Toggle)	reserved	reserved	reserved	Bit 7 of Command ID	Bit 6 of Command ID	Bit 5 of Command ID	Bit 4 of Command ID	Bit 3 of Command ID	Bit 2 of Command ID	Bit 1 of Command ID	Bit 0 of Command ID

Command Value	
Direction	Master ➟Slave
Size	1 word
Range	see table below
Unit	see table below

The table 3-1, “Overview of command module commands,” on page 40 describes the commands that can be executed with this module. All commands are started by altering the **Start Command** bits and apply at the same time to all motors specified with bits **Motor A...D**. It is therefore possible with one command to alter the **P** value of all motors.

The commands **Redefine Position** and **Move Home Position** freeze the position set with the **Set Position** module. This is necessary to prevent unwanted position jumps. With the command **Unlock Set Position** the position update is enabled again. The correct use of this command accordingly comprises the following steps:

1. Perform the **Redefine Position** command. With this command a new value is allocated to the actual position.
2. Adapt the target position transmitted with the **Set Position** command.
3. Call the command **Unlock Set Position** to enable the position setting again.

Command	ID HEX	Value from ... to	Unit	Description
No Command	0x00	-	-	No command is executed.
Redefine Position	0x01	-32256 … +32256	19.53125µm or 1/8 Step	Redefines the actual position and sets the demand position to the same value. This command freezes the position selected with the **Set Position** module. It can be enabled again with the **Unlock Set Position**.
Move Home Position	0x02	-32256 … +32256	19.53125µm or 1/8 Step	Shifts the reference position by the specified value. The demand position is shifted as well so that the motor does not move. This command freezes the position updated with the **Set Position** module. It can be enabled again with the **Unlock Set Position**.
Unlock Set Position	0x03	-	-	Enables target position updated with the **Set Position** module. This command is needed only after the **Redefine Position** and **Move Home Position** commands.
Set Demand Position to Actual Position	0x04			The Demand position is set to the actual position of the motor.
Set P	0x10	0 … 32640	0.00234 A/mm	Set **P** value of controller.
Set I	0x11	0 … 32640	0.0457 A/(mm*s)	Set **I** value of controller.
Set D	0x12	0 … 32640	0.015 A/(m/s)	Set **D** value of controller.
Set FF Friction	0x13	0 … 255	0.0234 A	Set **FF Friction** value of controller.
Set FF Acceleration	0x14	0 … 32640	0.1 mA/(m/s$^{2)}$	Set **FF Acceleration** value of controller.
Set FF Deceleration	0x15	0 … 32640	0.1 mA/(m/s$^{2)}$	Set **FF Deceleration** value of controller.
Set Current Offset	0x16	-256 … 256	23.4 mA	Set the **Current Offset** value of controller.

Table 3-1: Overview of command module commands

Control/Status

The **Control/Status** data module transmits the control word to the *LinMot*® servo controller and reads back the status word from the servo controller. It must be setup as first module. All subsequent modules relate to motor A till the **Next Drive** module follows. If the **Next Drive** module is not supported by the setup software, as an alternative the **Control/Status** module may be used again to introduce a new motor. But only the data in the first **Control/Status** will be evaluated.

Module: Control/Status	
Control	
Direction	Master ➟Slave
Size	1 word
Structure	

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	FREEZE Drive D	FREEZE Drive C	FREEZE Drive B	FREEZE Drive A	Trig In 4 (Drive D)	Trig In 3 (Drive C)	Trig In 2 (Drive B)	Trig In 1 (Drive A)	reserved	reserved	reserved	FREEZE ALL Req.	INIT Request	STOP Request	RUN Request	reserved

Status	
Direction	Slave ➟Master
Size	1 word
Structure	

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	In Position Motor D	In Position Motor C	In Position Motor B	In Position Motor A	Curve Done D (Toggle)	Curve Done C (Toggle)	Curve Done B (Toggle)	Curve Done A (Toggle)	WARNING Pending	ERROR Pending	CMD Executed (Toggle)	DISABLE State	INIT State	ERROR State	RUN State	INIT Done

The control word determines the state into which the servo controller has to go, and is sent from master to slave. The individual bits have the following meaning:

RUN Request	Requests change to **RUN** state
STOP Request	Requests change to **STOP** state
INIT Request	Requests change to **INIT** state
FREEZE ALL Req.	Requests change to **FREEZE** state for all Motors

These signals correspond to those of the AT/MT servo controller. Thus for example the change to the initialization mode is requested by **INIT Request**. All states are described in detail in the user manual in chapter 4.2 from page 4-6.

The signals **Trig In 1** to **Trig In 4** serve to initialize the motors in the initializing modes **Trig Move Out** or **Trig Move In**.

Trig In 1 (motor A)	Trigger signal for motor A
Trig In 2 (motor B)	Trigger signal for motor B
Trig In 3 (motor C)	Trigger signal for motor C
Trig In 4 (motor D)	Trigger signal for motor D

With the **FREEZE DRIVE X** signals, single motors can be interrupted within their movement.

FREEZE Drive A	FREEZE-Signal of Motor A
FREEZE Drive B	FREEZE-Signal of Motor B
FREEZE Drive C	FREEZE-Signal of Motor C
FREEZE Drive D	FREEZE-Signal of Motor D

The state in which the servo controller is signalled back by the status word. The individual bits have the following significance:

RUN State	Servo controller in **RUN** state
ERROR State	Servo controller in **ERROR** state
INIT State	Servo controller in **INIT** state
DISABLE State	Servo controller in **DISABLE** state
INIT Done	All motors have been initialized

The **CMD Executed** bit changes its polarity at every command executed with the **Command** module.

CMD Executed	This bit changes its polarity with every command executed.

The two signals **ERROR Pending** and **WARNING Pending** indicate whether there is an error or a warning present.

ERROR Pending	There is an error.
WARNING Pending	There is an warning.

The four bits **Curve Done A..D** indicate whether a motion profile commenced with the **Run Curve** module has been completed already. The motion profiles are always started by toggling a bit in the **Run Curve** module. As soon as the motion profile is ended the **Curve Done A..D** assumes the same value as the bit in the **Run Curve** module. In this way it is possible to determine any time whether a motion profile is still being executed or has been completed.

Curve Done A	Motion profile ended on motor A
Curve Done B	Motion profile ended on motor B
Curve Done C	Motion profile ended on motor C
Curve Done D	Motion profile ended on motor D

The four bits **In Position A..D** indicate whether the motor, after a new position has been set or a motion profile has been executed, has reached a defined range around the target point. The limits of this range can be set with the

parameters **In Position +** and **In Position -** in the directory **\Drives\Drive X\Position Monitoring**.

In Position Drive A	Motor A has reached the target point
In Position Drive B	Motor B has reached the target point
In Position Drive C	Motor C has reached the target point
In Position Drive D	Motor D has reached the target point

Get Position

This module serves to transmit the actual position of the motor from the *LinMot*® servo controller to the PLC.

Module: Get Position	
Actual Position	
Direction	Slave ➟Master
Size	1 word
Range	-32'256 to +32'256; (32'767 on invalid positions)
Unit	Motor type: Unit *LinMot*®: 19.53125µm Stepper: 1/8Step Solenoid: 23.438mA

Get Current

This module serves to transmit the actual current of the motor from the *LinMot*® servo controller to the PLC.

Module: Get Current	
Actual Position	
Direction	Slave ➟Master
Size	1 word
Range	-256 to +256
Unit	23.438mA

Max. Acceleration

This module sets the maximum acceleration of the motor.

Module: Max. Acceleration	
Max. Acceleration	
Direction	Master ➟Slave
Size	1 word
Range	1 to 1536
Unit	Motor type: Unit *LinMot*®: 238.419mm/s^2 Stepper: 47.6836Steps/s^2 Solenoid: -

Max. Speed

This module sets the maximum speed of the motor

Module: Max. Speed	
Max. Speed	
Direction	Master ➟Slave
Size	1 word
Range	6 to 24576
Unit	

Motor type	Unit
LinMot®	190.735 µm/s
Stepper	0.081469 Steps/s
Solenoid	-

Max. Current

This module sets the maximum current (power) of the motor

Module: Max. Current	
Max. Current	
Direction	Master ➟Slave
Size	1 word
Range	0 to 255
Unit	23.438 mA

Next Drive

This module introduces the next motor when setup. Individual setup software packages do not support this module, because it transmits no useful data. If this is the case the **Control/Status** module may be used instead.

Module: Next Drive	
Next Drive	
Direction	Master ➟Slave
Size	0 words

Run Curve

This module serves to run a motion profile on the servo controller.
While the “Continuous” Flag is set, the motion profile is repeated continuously.
If the “Delayed” flag is set the motion profile will not be started as long as another profile is running.

With the “Incremental” flag, the actual Set Position is used as “Curve Offset”.
When this flag is used, the module “Curve Offset” should not be configured.

Module: Run Curve	
Curve	
Direction	Master ➟Slave
Size	1 word
Structure	

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Run Curve (Toggle)	Continuous (from Rel.1.3.9)	Delayed (from Rel.1.3.9)	Incremental (from Rel.1.3.9)	ME Mode (from Rel. 1.3.10)*	reserved	reserved	reserved	reserved	reserved	Bit 5 of Curve Number	Bit 4 of Curve Number	Bit 3 of Curve Number	Bit 2 of Curve Number	Bit 1 of Curve Number	Bit 0 of Curve Number

**) The “ME Mode” flag is only valid for MP software (Master Encoder with Profibus) and if it is set the motion profile will run in master encoder mode.*

Set Curve Speed

This module serves to set the desired speed of motion profiles. If the maximum value is set the motion profile is run as quickly as it was created. With lower values the speed drops linearly.

Module: Set Curve Speed	
Curve Speed	
Direction	Master ➟Slave
Size	1 word
Range	0 to 4096
Unit	0.0244% of maximum speed

Set Curve Amplitude

This module serves to set the desired amplitude of motion profiles. The maximum value (4096) is equal to the scale factor 100%. With this value the amplitude of the motion profile is as big as it was defined in the curve creator.

Module: Set Curve Amplitude	
Curve Amplitude	
Direction	Master ➟Slave
Size	1 word
Range	0 to 4096
Unit	0.0244% of maximum amplitude

Set Curve Offset

This module serves to set the desired offset of the motion profiles.

Module: Set Curve Offset	
Curve Offset	
Direction	Master ➟Slave
Size	1 word
Range	-32256 to +32256
Unit	Motor type: *LinMot®* → Unit: 19.53125µm Motor type: Stepper → Unit: 1/8 Step Motor type: Solenoid → Unit: 23.438mA

Set Position

This module serves to transmit the desired target position of the motor to the *LinMot®* servo controller.

Module: Set Position	
Demand Position	
Direction	Master ➟Slave
Size	1 word
Range	-32256 to +32256
Unit	Motor type: *LinMot®* → Unit: 19.53125µm Motor type: Stepper → Unit: 1/8 Step Motor type: Solenoid → Unit: 23.438mA

3.9 Diagnosis

PROFIBUS-DP is prepared for diagnosing equipment with messages in clear text. The *LinMot*® servo controller and most of the setup software packages support this. In the diagnosis messages any warnings and errors of a DP slave present appear in clear text. If the diagnose is to be evaluated with a PLC program, recourse may be had to the description of the diagnosis data in the GSD file. There the meanings of all bits are defined.

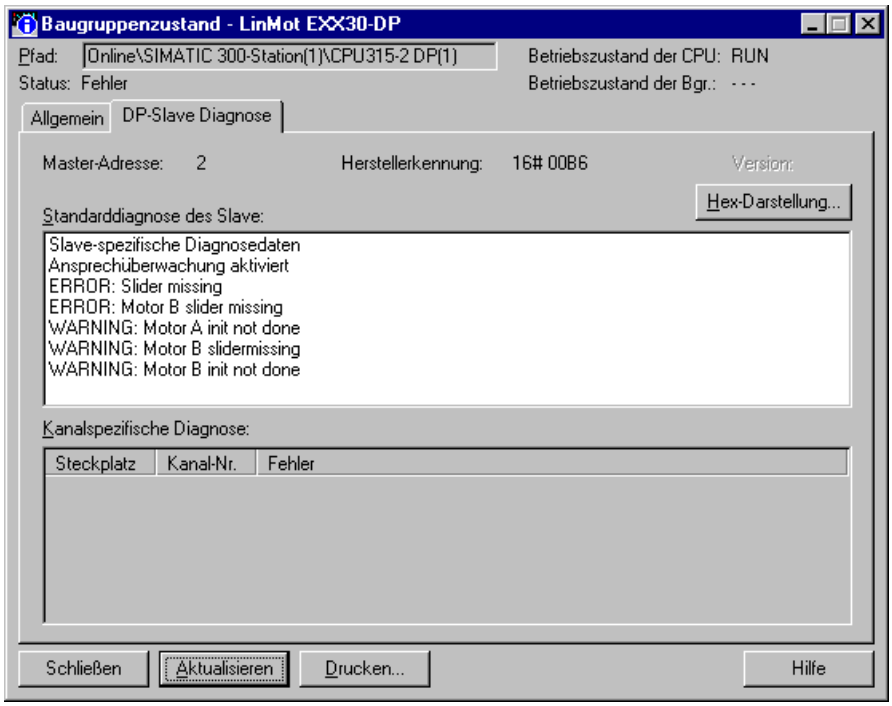


Figure 3-4: Diagnosis message of *LinMot*® servo controller

The diagnostic telegram consists of 28 Byte. In the following tables, the mapping of diagnostic data to the warnings and errors of the *LinMot*® System are shown:

Byte No.	Meaning
0-5	According the PROFIBUS standard
6-7	Header and Padding
8-9	System Errors
10-11	Motor A Errors
12-13	Motor B Errors
14-15	Motor C Errors
16-17	Motor D Errors
18-19	System Warnings
20-21	Motor A Warnings
22-23	Motor B Warnings
24-25	Motor C Warnings
26-27	Motor D Warnings

System and Motor Errors		
Bit No	**Error**	**Cause**
0	Motor too hot calculated	The calculated temperature of the motor is too hot
1	Motor too hot sensor	The measured temperature of the motor is too hot
2	Following Error	Following Error
3	Slider Missing	- *LinMot*: No slider in the stator or the slider is extracted too much - *External Sensor:* The distance between the sensor head and the magnetic band is too big, or the magnetic band is damaged
4	Slave Error	An attached Booster or Gantry slave motor, has an error. The Type of error can be found in the diagnostic data of the slave motor.
5	Init failed	Error during the Homing procedure
6	Motor Type mismatch	The type of the connected motor is not identical with the configured one
7	Curve Missing	The started curve is not stored in the Controller
8	Reserved	
9	DCLV Power too low	The DC Link Voltage of the power board is too low
10	DCLV Power too high	The DC Link Voltage of the power board is too high (e.g. extensive breaking)
11	DCLV Signal too low	The Power Supply Voltage of the micro controller board is too low
12	DCLV Signal too high	The Power Supply Voltage of the micro controller board is too high
13	Electronic Fault	E1000 series: The Servo Controller is too hot, or the short-circuit detection has triggered. E100 series: The Servo Controller is too hot.
14	Reserved	
15	Application Error	DP specific Error (e.g. data out of Range)

System Warnings		
Bit No	**Warning**	**Cause**
0	Motor too hot calculated	The calculated temperature of the motor is hot
1	Motor too hot sensor	The measured temperature of the motor is hot
2	Following Error	Following Error
3	Slider Missing	- *LinMot*: No slider in the stator or the slider is extracted too much - *External Sensor:* The distance between the sensor head and the magnetic band is too big, or the magnetic band is damaged
4	Reserved	
5	Reserved	
6	Init not done	The Homing has not finished yet
7	Reserved	
8	Low Free Capacity	System Resources low
9	DCLV Power small	The DC Link Voltage of the power board is low
10	DCLV Power high	The DC Link Voltage of the power board is high (e.g. extensive breaking)
11	DCLV Signal small	The Power Supply Voltage of the micro controller board is low
12	DCLV Signal high	The Power Supply Voltage of the m micro controller board is high
13	Electronic Fault	The Servo Controller is hot,
14	Emergency Stop	The servo controller is in the emergency stop state.
15	Reserved	

Motor Warnings		
Bit No	**Warning**	**Cause**
0	Motor hot calculated	The calculated temperature of the motor is hot
1	Motor hot sensor	The measured temperature of the motor is hot
2	Following Error	Following Error
3	Slider Missing	- *LinMot*: No slider in the stator or the slider is extracted too much - *External Sensor:* The distance between the sensor head and the magnetic band is too big, or the magnetic band is damaged
4	Slave warning	An attached Booster or Gantry slave motor, has an warning. The Type of warning can be found in the diagnostic data of the slave motor.
5	Reserved	
6	Init not done	The Homing has not finished yet
7	Reserved	
8	Mot not in Pos Range	The motor is not within the “Pos Range” (configured with LinMot Talk)
9	DCLV Power small	The DC Link Voltage of the power board is low
10	DCLV Power high	The DC Link Voltage of the power board is high (e.g. extensive breaking)
11	DCLV Signal small	The Power Supply Voltage of the micro controller board is low
12	DCLV Signal high	The Power Supply Voltage of the micro controller board is high
13	Electronic Fault	The Servo Controller is hot
14	Reserved	
15	Reserved	

3.10 Trouble shooting / remedying faults

PROFIBUS-DP is a very robust industrial bus, consequently most problems when commissioning can usually be traced to faulty cabling or configuring and not to defective equipment. With communication problems the bus cabling should be checked first, and then the configuring. The following tips have proved useful in practice.

Clearing the PLC

It is advisable to clear the PLC completely every time when altering the configuration or software.

Servo controller will not go on-line

Check DP address: It is adjusted with the two rotary switches on the front of the servo controller. Important: The address is entered hexadecimal.

Check the cabling: Only the two connectors at the end of the bus may be terminated. Because the bus is terminated actively, for correct termination the equipment at the end of the bus must be switched on so that the connectors have the voltage for termination. Check possibly whether the equipment in question also have a voltage supply for the external termination. In case of doubt place *LinMot®* servo controller at one end of the bus and the PLC at the other.

Stub lines are no longer admissible with 12MBit/s.

Check version of GSD data file.

Motor not initialized

If data modules are configured for maximum acceleration, maximum speed or maximum current, they must be preloaded with suitable values so that the motor may be initialized.

Byte order

Unfortunately the various PLC makers use different byte order definitions in their equipment. One of them gives the smallest address to the least significant byte (LSB) in a word, while another does just the opposite. To circumvent this problem, with *LinMot®* servo controller the byte order can be adjusted by means of parameters. A wrongly configured byte order may result in the lower and upper bytes being switched round in a word.

Long cycle time

With slow PLC controllers it is advisable to cut out the high-priority diagnosis on the *LinMot®* servo controller, in order to shorten the maximum cycle time[1] . In case of error there will then be no high-priority interrupt triggered on the PLC. The error must then be handled in the cyclic OB.

To obtain a short cycle time it is advisable to map the *LinMot®* servo controller into the process image and not perform subsequent peripheral accessing.

Depending on the application and control it may make sense to map the *LinMot®* servo controller into their own part process image. This eliminates cyclic process image updating.

1) Here the cycle time on the PLC is meant. The bus cycle time on PROFIBUS is independent of the PLC cycle time in most cases.

3.11 Interfaces

The PROFIBUS servo controller have the following interfaces:

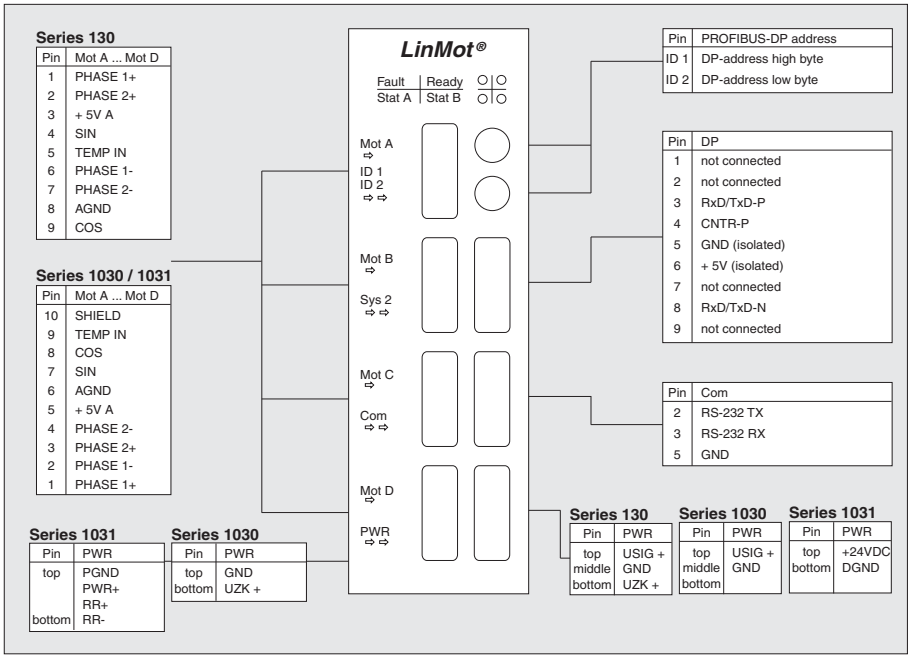


Figure 3-5: Pin assignment of PROFIBUS servo controller

4. External position sensing

The repeatable position accuracy of the linear motors of the *LinMot*® P series with integrated measuring system can be increased to 10µm or better with an external position sensing. The very high linearity of the external sensor tape enables the deployment of linear motors in applications where a very high accuracy is demanded.

The *LinMot*® servo controller support two types of external position sensors links: Sine/Cosine and A/B (incremental).

4.1 Sine/Cosine sensors

Principle

The sine/cosine position sensor consists of a sensing head and a magnetic scale strip. When the sensing head is moved over the scale strip, a sine and a cosine signal are given at the output. From these signals the *LinMot*® servo controller calculates a position signal, which can be used to position the motors.

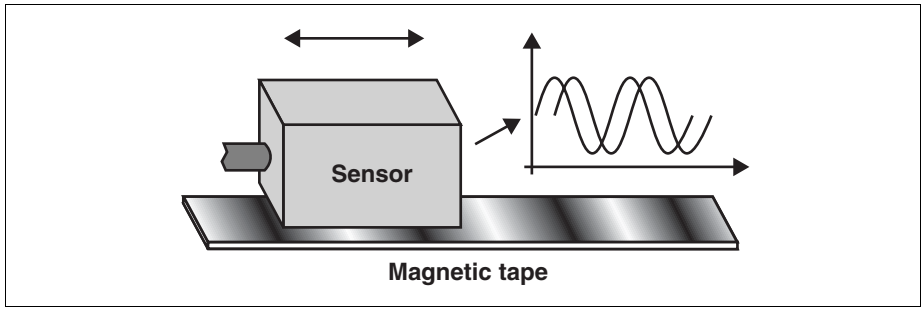


Figure 4-1: Principle of the sin/cos position sensing

Sensing connection

The external sensing is connected to an unused motor channel. The motor that obtains its position via the external sensing must always be connected to the following channel.

Example: If the external position sensing is connected to the motor channel A the motor has to be connected to the channel B. If the external position sensing is connected to the motor channel C the motor has to be connected to the channel D.

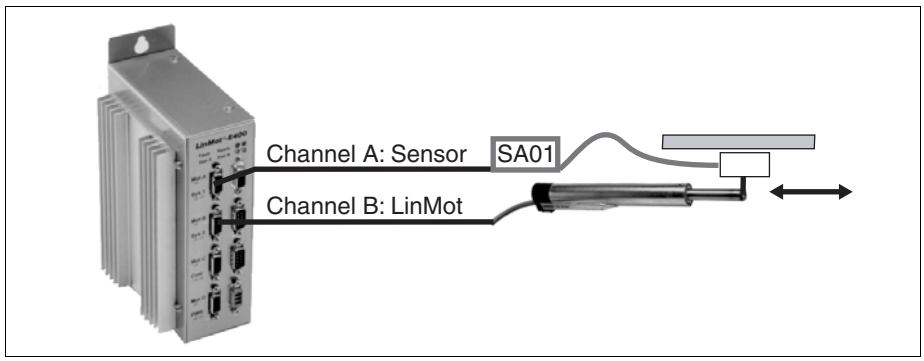


Figure 4-2: Position sensing connection

Operation with *LinMot*® position sensors MS01-1000/P and MS01-5000/P

If operating with the *LinMot*® position sensors MS01-1000/P or MS01-5000/P the sensor amplifier SA01-44/D is needed. It performs the amplification of the differential sine and cosine signals. The following diagram shows how to connect the position sensor.

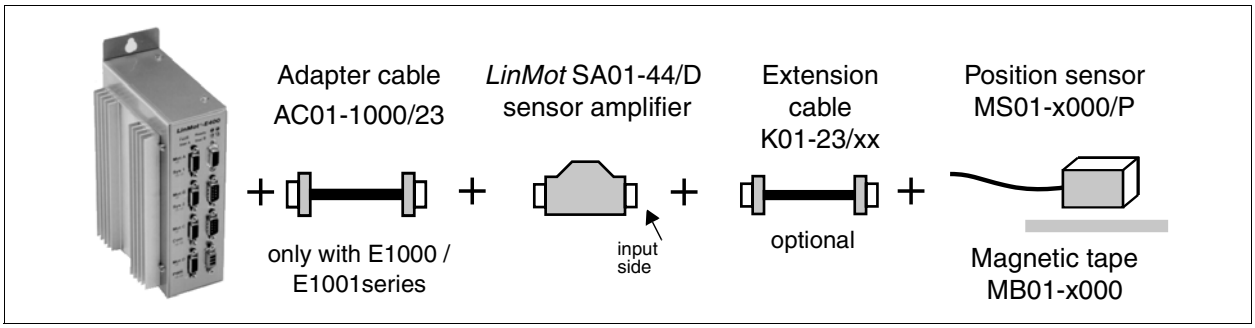


Figure 4-3: Connecting a position sensor using the *LinMot*® sensor amplifier

The table below shows the pin out on the input side of the sensing amplifier. Both sine and cosine signals have differential inputs, making them immune to electromagnetic interference.

Name of signal	*LinMot*® sensing amplifier	Description
GND	Pin 8	Earth
+5V	Pin 7 (output)	+5 V supply
Sensor Sin+	Pin 2 (input)	Differential sine input Amplitude: ±50 mV Common mode range: 0…5V
Sensor Sin-	Pin 1 (input)	
Sensor Cos+	Pin 5 (input	Differential cosine input Amplitude: ±50 mV Common mode range: 0…5V
Sensor Cos-	Pin 4 (input)	

Operating without *LinMot*® sensor amplifier

Operation without *LinMot*® sensor amplifier is inadvisable. If position sensors from other manufacturers are used the sine and cosine signals must have an offset voltage of +2.5V and an amplitude of max. ±2.3 V. Pin out is as follows:

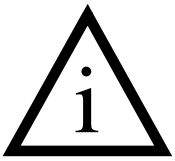
Signal Name	*LinMot*® E100 Series	*LinMot*® E1000/ E1001 Series	Description
GND	Pin 8	Pin 6	Earth
$+5V_{out}$	Pin 3	Pin 5	5 Volt Output Max. 50 mA
Sensor Sin	Pin 4	Pin 7	Sine input Offset: $+5V_{out}$ / 2 Amplitude: ±2.3 Volt
Sensor Cos	Pin 9	Pin 8	Cosine input Offset: $+5V_{out}$ / 2 Amplitude: ±2.3 Volt

Resolution

The table below correlates the resolution and stroke range of *LinMot®* motors.

Resolution and stroke range		
Pole pitch	**Resolution**	**Stroke range**
1mm	20µm	1260mm
	10µm	630mm
	5µm	315mm
	2.5µm	157.5mm
	1.25µm	78.75mm
5mm	20µm	1260mm
	10µm	630mm
	5µm	315mm

LinMot® Talk



If the external position sensing is employed and a resolution is other than **20µm**, allowance must be made for this in the following parameters.

- Home / Check, Init / Initial Position
- Minimal/Maximal Position
- 0V / 10V / ’0’ / ’1’ Position
- all parameters in the position monitoring directory
- all parameters (Position, speed and acceleration) for generating motion profiles in the **Curve Inspector**.
- all position values (but not speed and acceleration values) in: multitrigger tables, the ASCII protocol and the PROFIBUS

All these parameters must be converted by the following formula:

$$\mathbf{Value_{new} = Value_{real} * Factor}$$

The factor has the following value:

Resolution	Factor
10µm	2
5µm	4
2.5µm	8
1.25µm	16

Example

The resolution has been set to **5µm**. If the **Home Position** parameter is now to be adjusted to 30mm, a value of 120mm must be entered instead.

4.2 A/B sensors

Principle

The A/B position sensor consists of a sensing head and a magnetic or other scale strip. When the sensing head is moved over the scale strip, it will generate two digital output signals A and B, which are phase shifted by 90 degrees. The signals A and B are feed into an encoder link of the master encoder over RS422. From these signals the *LinMot*® servo controller calculates a position signal, which can be used to position the motors.

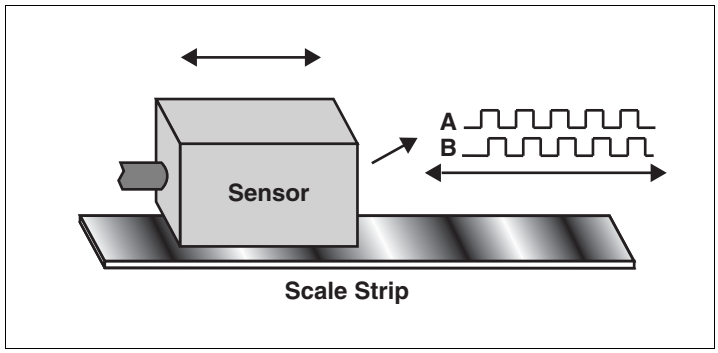


Figure 4-4: Principle of the A/B position sensing

Sensing connection

The external A/B sensing is connected to an encoder link of the master encoder module. See “Addendum Master Encoder”. The signal type of the encoder link is RS422 and the encoder can be freely assigned to the motor by the software.

On version 2 controllers, the master encoder functionality is provide with add on modules, which have the following links:

	ME01-01/08	ME01-02/08
Link A	**ME / Pos Sensor**	**ME / Pos Sensor**
Link B	**loop through only**	**Pos Sensor**

On version 3 controllers, the master encoder functionality is alsready included with the option -ME. So no add on modules are necessary.

Parameters

When a motor gets its actual position from a A/B sensing, the only parameter to be set is under Drive X\Advanced\Position Sensor. The resolution of the encoder can be modified by selecting the decode mode.

There are the following possibilities:

A/B sensor specific items in \Drives\Drive X\Advanced\Position Sensor	
AB Enc1 1X	Position from Encoder Link 1, Decode mode 1x
AB Enc1 2X	Position from Encoder Link 1, Decode mode 2x
AB Enc1 4X	Position from Encoder Link 1, Decode mode 4x
AB Enc2 1X	Position from Encoder Link 2, Decode mode 1x
AB Enc2 2X	Position from Encoder Link 2, Decode mode 2x
AB Enc2 1X	Position from Encoder Link 2, Decode mode 4x

5. Master/Slave Modes

As a *LinMot*® controller can serve up to four motors, several motor can work together as one logical axis. Two different master/slave modes are supported: master/booster and master/gantry slave.

5.1 Master/Booster operation

Master/Booster operation enables the force available for a movement to be increased by putting motors in series or parallel. One motor must be defined as master and up to three others as boosters. In addition the sliders of the motors are coupled together mechanically.

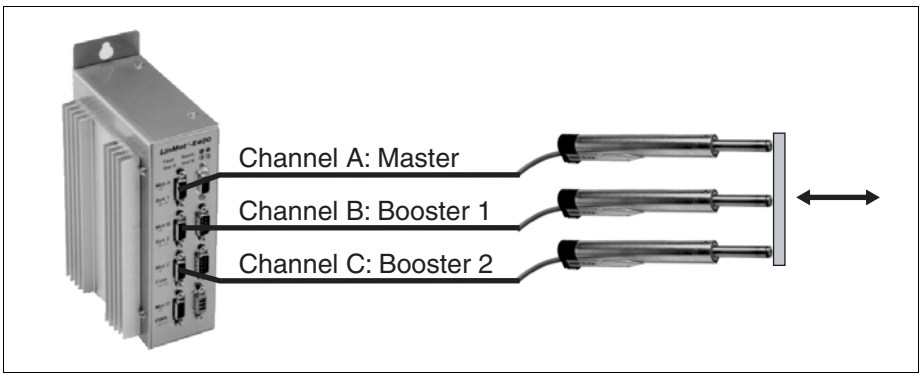


Figure 5-1: Master/Booster operation of *LinMot*® motors

Principle

In operation the position is set for the master motor. The current calculated by the position controller on the master motor is now set for both the master and the booster motors.

Parameters

When configuring the motors the master must be defined before the boosters. Only the following parameters must be defined on booster motors:

- motor type (P0x-23, P0x-37, P0x-48), same type as master
- 2 or 3 Amps (only with E100 series), same value as master
- commutation
- error handling

Operating Modes

A booster motor may exert its force either in the same direction as the master or opposed to it. This can be chosen in the **Master / Booster** directory. In the **Booster parallel** setting the booster motor must point in the direction of the master, in the **Booster reverse** setting in the opposite direction to the master. See also Figure 5-2, “Booster operating modes” below.

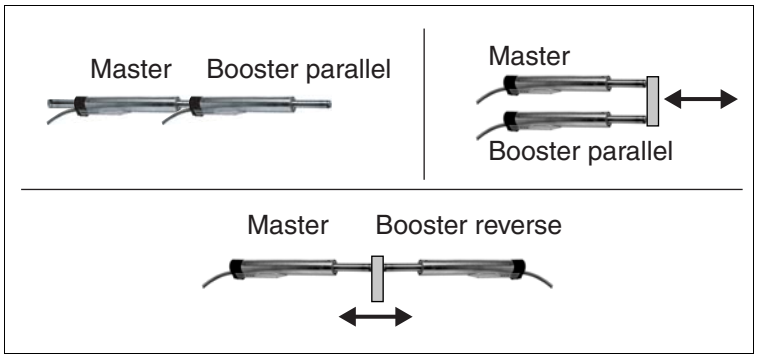


Figure 5-2: Booster operating modes

5.2 Master/Gantry operation

Like Master/Booster operation the Master/Gantry operation enables working of two or more motors together by commanding just the master motor. But in contrast to the Booster motor, the gantry slave motor is completely position controlled, it only has copied the commanded positions from the master. This enables another type of mechanical constructions where the two motors are placed far from each other and the mechanical coupling can be weak (or even inexistent). There can be configured up to three gantry slave on the same controller. It is also possible to combine external sensors with the gantry master and slaves.

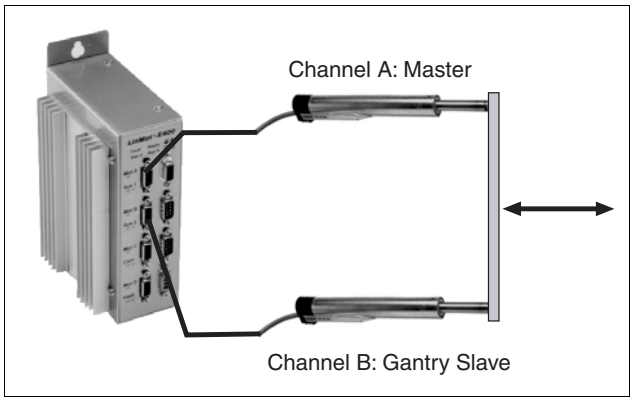


Figure 5-3: Gantry operation

Principle

In operation the position is set for the master motor and will be copied automatically to the slave. The motors are position controlled independently of each other.

Parameters

When configuring the motors the master must be defined before the slaves. The slaves must be completely configured except for the run mode. A slave is defined by selecting “Gantry Slave parallel” under “DriveX\Advanced\Master / Booster\”.

6. Parameters

6.1 Introduction

All servo controllers of the *LinMot*® family may be configured for the application by means of parameters. The configuration is stored on the servo controller in the nonvolatile EEPROM.

The parameters are grouped hierarchically (tree structure) and may be edited simply with the **Parameter Inspector**. All parameters are listed and explained in tabular form below.

PARAMETER TABLES In these tables all parameters are explained. Each table describes a directory or a parameter which may have various predefined values.

A —	\Drives\Drive X\...		
B —	**Parameter name**	Explanation	— D
C —	L		
	...	...	
	...	...	

A The parameters described are in this directory.

B The names of the parameters.

C Attributes. See also table 6-1, meanings of the attributes.

D Description of parameter or predefined values.

All **parameters** and **directories** are printed in **bold type** in this section so that they stand out. Every parameter may have additional attributes. The possible attributes are explained in the table below.

Attribute	Meaning
L	This symbol signifies live parameter. They may be altered while the motors are operating.
R	This symbol signifies parameters that are write-protected and cannot be altered by the user.
👁	This symbol indicates that the parameter is visible only under certain circumstances. A footnote at the foot of the table states when this is so.

Table 6-1: Meaning of the attributes

To obtain a better overview, associated parameters are described in separate sections.

System parameters These include password protection, system-related error handling, starting behavior, system time and firmware version information.

Drive parameters In the **Drives** directory up to four actuators (Drive A, Drive B, Drive C, Drive D) may be configured independently of each other. The motor parameters define the behavior of the actuators concerned. This is determined by the drive type,

the initialization, setpoint generation, positioning monitoring, control parameters and the drive-related error handling. The parameter for configuring the linear motors, stepper motors, solenoids and external position sensors are described in separate sections.

MT parameters

The **Multi Trigger** directory serves to configure the MT servo controller. It is visible only if a MT servo controller is connected and if the **Command Interface** parameter in the **System** directory has the value **MT**.

PROFIBUS parameters

In the **PROFIBUS** directory settings for the PROFIBUS servo controller can be made. They are visible only with the servo controller having a PROFIBUS interface (e.g. E430-DP).

Where are the parameter descriptions

The table below shows the sections describing the individual parameters.

6.2 Global parameters

Global parameters define the global behavior of the system. This includes password treatment, error handling, startup behavior, in/output configuration, time and firmware information.

Device information

Device

The parameters in the **Device** directory provide information on the system hardware.

\System\Info\Hardware\Device R	
Type	Gives information on servo controller type
Serial No High	Top three digits of serial number
Serial No Low	Bottom three digits of serial number
Serial Number	Serial number as a string
Article Number	Article number as a string

Memory

The parameters in the **Memory** directory provide information on the storage of the servo controller.

\System\Info\Hardware\Memory R	
Flash Type	FLASH-EPROM type used
EEPROM Type	EEPROM type used
RAM Type	RAM type used

Software

The parameters in the **Software** directory describe the software installed in the servo controller.

\System\Info\Software R	
Release	Software release
Monitor	Monitor release
Base	Firmware version
Application	Application software version
Application 2	Application 2 software version
Tree Type	Parameter tree type
Tree Version	Parameter tree version

ID Switch Position

This parameter shows the ID switch position.

\System\Info	
ID Switch Position L	This value displays the position of the two hexadecimal rotary ID switches on the controller (on the front side if DP controller or version 3 controller, else on the bottom side), which are used to define the MAC-ID for bus interfaces.

Passwords

Passwords

In the **Passwords** directory the password for the servo controller can be set.

\System\Passwords	
User	The **User** parameter contains the currently used user password. It can be altered only at this place. No password is set when the servo controllers are supplied.

Error handling

In this subsection the global handling of errors is described. Here a distinction is made between warnings and faults. In most cases a warning is given first when an error occurs. If the higher-level system does not respond to the pending warning, the servo controller goes to the error state.

Warnings

A warning is given with an active digital output signal **WARNING OUT**. This is activated when a warning occurs. When a warning message is given, the system is still fully operational. All motors are controlled. A warning message enables the higher-level system to respond to the warning and carry out a controlled system shutdown (e.g. bring it into an emergency stop position). When a warning is given the servo controller is always still in the **RUN** state.

Error

An error is signalled on the one hand by activating the digital output signal **ERROR OUT**. This is done when an error occurs. On the other hand an error leads to an immediate switch-off of all motors controlled by the servo controller. The servo controller is then in the **ERROR** state.

With fatal errors there is an instant jump into the error state.

Error Mask[1]

The selected entries determine which errors cause the firmware to go to the error status. The motors are stopped if there is an error.

\System\Error Handling\Error Mask R	
DCLV Power Too Low	Power supply voltage too low.
DCLV Power Too High	Power supply voltage too high.
DCLV Signal Too Low	Signal supply voltage too low.
DCLV Signal Too High	Signal supply voltage too high.
Electronic Fault	Heat sink of servo controller is too hot (over 70°C) or a short circuit has been detected on a motor phase.

Warn Mask[1]

In this directory all internal errors are selected that are to trigger a warning signal on the digital output.

\System\Error Handling\Warn Mask	
DCLV Power Low	Power supply voltage low.
DCLV Power High	Power supply voltage high.
DCLV Signal Low	Signal supply voltage low.
DCLV Signal High	Signal supply voltage high.
Electronic Fault	Heat sink of servo controller hot[1] (over 70°C) or a short circuit has been detected on a motor-phase.

1) As soon as the heat sink temperature is over 70°C the warning **Electronic Fault** is activated.Then after 5 seconds the error **Electronic Fault** is activated.

Msg Mask

The parameters in the **Msg Mask** directory specify when the **Msg Output** signal should be high (parameter selected) or low (parameter not selected).

1) The exact threshold values for the voltage monitoring are shown in table 6-2 on page 65.

Logging Mask

The parameters in the Logging Mask directory specify all internal error that should be stored by the servo controller. The **error log** is preserved during power failures.

\System\Error Handling\Msg Mask [1] **\System\Error Handling\Logging Mask**	
DCLV Power Too Low	The supply voltage for the power circuitry is too low.
DCLV Power Too High	The supply voltage for the power circuitry is too high.
DCLV Signal Too Low	The supply voltage for the power circuitry is too low.
DCLV Signal Too High	The supply voltage for the power circuitry is too high.
Electronic Fault	Heat sink of servo controller is too hot (over 70°C) or a short circuit has been detected on a motor phase.
Drive Type Mismatch	The connected actuator type does not correspond with the selected actuator type.
Curve Error	A reference motion profile cannot be found, or the desired profile is not compatible with the selected actuator type.
Slider Missing	The slider is missing from a motor, or the motor was not connected correctly.
Init Failed	The initialization process could not be completed successfully.
Drive Following Error	The following error of a motor is too big. The following error can be set separately for each motor.
Drive Too Hot Calculated	A motor was temporary overloaded. Possible causes are: the motor was blocked or overloaded (load mass too big, trajectory too fast, ...). If this parameter is unselected, it can be possible that an overheating of the motor, which is caused ba a shorttime overload, cannot be detected anymore. The motor may be damaged.
Drive Too Hot Sensor	A motor is too hot. It is overloaded and/or insufficiently cooled.

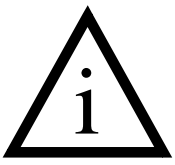
1) Not visible if parameter **MT** has been selected in the **\System\Command Interface** directory.

DCLV Monitoring

The warnings and errors in the voltage monitoring are generated according to the values set out below.

\System\Error Handling\DCLV Monitoring R	
Power Low Warn	A warning is given if this power supply voltage is understepped.
Power High Warn	A warning is given if this power supply voltage is exceeded.
Power Low Error	An error is signalled if this power supply voltage is understepped.
Power High Error	An error is signalled if this power supply voltage is exceeded.
Signal Low Warn	A warning is given if this signal supply voltage is understepped.
Signal High Warn	A warning is given if this signal supply voltage is exceeded.
Signal Low Error	An error is signalled if this signal supply voltage is understepped.
Signal High Error	An error is signalled if this signal supply voltage is overstepped.

By default, a warning is given if the nominal voltage is exceeded by 6% or understepped by 12%. An error leading inevitably to the servo controller being run down is signalled if the nominal voltage is exceeded by 12% or understepped by 24%.



We recommend checking the voltage supply before and during the first commissioning, ascertaining above all whether the connected power supply has sufficient voltage and current.

		Nominal	Default Warning threshold[V]	Default Error threshold[V]
Signal voltage 24-48V	Min.	24V	21.12V	18.24V
	Max.	48V	50.88V	53.76V
Power voltage 24-48V	Min.	24V	21.12V	18.24V
	Max.	48V	50.88V	53.76V
Power voltage 48-72V	Min.	48V	42.24V	36.48V
	Max.	72V	76.32V	77.93V

Table 6-2: Default thresholds for supply voltage monitoring for version 2 controllers

		Nominal	Default Warning threshold[V]	Default Error threshold[V]
Signal voltage 24-48V	Min.	24V	21.12V	18.25V
	Max.	48V	50.86V	53.73V
Power voltage 48-72V	Min.	48V	42.24V	36.50V
	Max.	80V	90.23V	92.29V

Table 6-3: Default thresholds for supply voltage monitoring for version 3 controllers

Start-up behaviour

Startup Mode

In the **Startup Mode** directory there is an on/off parameter with which the startup behavior of the firmware can be determined.

\System\Startup Mode	
Auto Start	If this parameter is set the servo controller jumps automatically after powerup into the **INIT** state and then to **RUN**, provided no error and warning occurred.
Init Together	If this parameter is set all the configured motors will start their initialization at the same time. If this parameter is cleared, motor A will be initializing first, then motor B, C and D.

I/O configuration

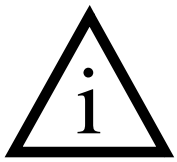
IO Configuration

The parameters in this directory define which input and output signals of the standard interface are to be read in or written out.

\System\IO Configuration	
Run Input	These parameters determine whether the signals from the interface are read in.
Init Input	
Freeze Input	
Emerg Stop Input	
Analog / Trig Drive A	These parameters determine whether the trigger signals are read in the states **DRIVE INIT** and **RUN**.
Analog / Trig Drive B	
Analog / Trig Drive C	
Analog / Trig Drive D	
Error Output	These parameters determine whether the corresponding signals are written to the interface.
Warn Output	
Pos Error Output [1]	
Msg Output [1]	

1) Not visible if parameter **MT** has been selected in the **\System\Command Interface** directory.

Normally only the inputs and outputs needed or processed by the higher level control system should be selected.



Command interface

Command Interface

Listed under **Command Interface** is the interface from which the commands for the servo controller are given. Only one interface at a time may be activated.

\System\Command Interface	
AT	The commands are given via the AT interface. This is the default setting.
MT	The commands are given via the MT interface. This interface may be selected only if a MT servo controller is connected, otherwise an error will be signalled upon start-up.
ASCII RS232	The commands are given via the RS232 interface assisted by the ASCII protocol. The LinMot-Talk protocol for configuration and debugging is still available over the RS232 link.
ASCII RS485	The commands are given via the RS485 interface assisted by the ASCII protocol. The LinMot-Talk protocol for configuration and debugging is still available over the RS485 link.
Application	The commands are given via a special application software, which can be programmed for special applications by NTI Ltd.

System time

Time

In this directory the system time can be recalled, broken down into operating hours and seconds. If the servo controller is stopped via the **Stop** key on the control panel, the system time is no longer updated till the next start.

\System\Time	
Hours	Operating hours
Seconds	Number of operating seconds (0…3600s)

Regeneration Resistor

The regeneration resistor's switching hysteresys can be configured with these parameters. This function is only available on version 3 controllers. On the screw terminals MOT SUPPLY can be connected an external regeneration resistor, which will be powered on demand over the internal MOSFET.

\System\Regeneration Resistor L	
Switch Off Voltage	If the voltage of the power supply goes below this limit, the regeneration resistor will be turned off (MOSFET becomes high impedant).
Switch On Voltage	If the voltage of the power supply rises above this limit, the regeneration resistor will be turned on (MOSFET becomes low impedant).

6.3 Motor parameters

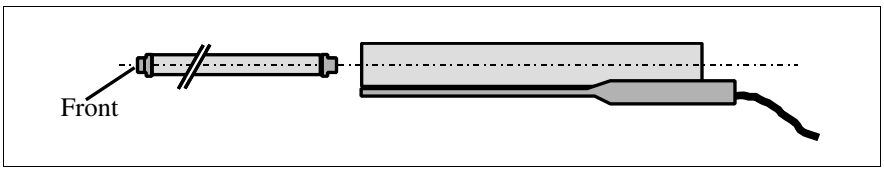
The actuator interfaces **Drive A** to **Drive D** all have the same parameters. The parameters described below are consequently the same for all four actuator interfaces

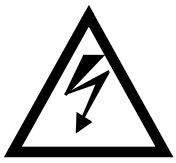
The drive parameters are divided into subsections giving the definition of the drive type, initialization, setpoint generation, position monitoring, control and error handling. Different subdirectories appear depending on the chosen motor type.

Type

These parameters define the connected actuator type. The following actuators may be selected:

\Drives\Drive X\Type	
No Drive	This type must be selected if no actuator is connected or the actuator is to be cut out.
***LinMot*® P0x-23**	This type must be selected if a *LinMot*® P0x-23 drive is connected.
***LinMot*® P0x-23F**	This type must be selected if a *LinMot*® P0x-23F (fast) drive is connected.
***LinMot*® P0x-37**	This type must be selected if a *LinMot*® P0x-37 drive is connected.
***LinMot*® P0x-37 PL19**	This type must be selected if a *LinMot*® P0x-37 drive in combination with a high clearance slider is connected.
***LinMot*® P0x-37-HP**	This type must be selected if a *LinMot*® P0x-37-HP (high performance) drive is connected.
***LinMot*® P0x-37-HP PL19**	This type must be selected if a *LinMot*® P0x-37-HP (high performance) drive in combination with a high clearance slider is connected.
***LinMot*® P0x-37F**	This type must be selected if a *LinMot*® P0x-37F (fast) drive is connected.
***LinMot*® P0x-37F PL19**	This type must be selected if a *LinMot*® P0x-37F (fast) drive in combination with a high clearance slider is connected.
***LinMot*® P0x-48**	This type must be selected if a *LinMot*® P0x-48 drive is connected.
***LinMot*® P0x-48 PL27**	This type must be selected if a *LinMot*® P0x-48 drive in combination with a high clearance slider is connected.
Stepper	If a stepper motor is connected, this type must be selected. The stepper motor must be two-phase.
Magnet	This drive type is selected to control a solenoid with the servo controller.
Sin/Cos Position Sensor	If an external position sensor (sine/cosine type) is to be connected to this channel, this type must be selected. See also chapter “External position sensing” on page 53.





Make absolutely certain that the configuration matches the motor type being used. A wrong configuration can lead to damage of the connected drive (linear motor, stepper motor, solenoid).

The actuators **stepper motor** and **solenoid** normally do not have built-in temperature sensors, therefore they are not monitored for overload. The user is himself responsible that these drives are not damaged in operation due to overloading (overheating).

The parameters for the different actuator types are described in separate chapters:

Linear motor	chapter “Linear motor parameters” on page 71
Stepper motor	chapter “Stepper motor parameters” on page 86
Solenoid	chapter “Solenoid parameters” on page 95
Positions Sensor	chapter “Position sensing parameters” on page 99

6.4 Linear motor parameters

In this section all parameters needed for configuration *LinMot*® linear motors are explained. To make these parameters visible in the parameter inspector a *LinMot*® motor in the directory **\Drives\Drive X\Type** must be selected.

Master / Booster operation

Master / Booster

This parameter defines whether the connected motor is to be operated in the master, booster or gantry slave mode. A motor operated in the booster mode takes over most of its master parameters. Thus if **Drive B** is configured as booster it takes over adjustments from **Drive A.** A detailed explanation is given in chapter “There are the following possibilities:” on page 57.

If a drive is configured as a gantry slave, it will get the save motion commands but the position controller is done separately.

\Drives\Drive X\Advanced\Master / Booster	
Master	The connected motor is master.
Booster parallel	The connected motor is a booster running with the master.
Booster reverse	The connected motor is a booster running in opposition to the master.
Gantry Slave parallel	The connected motor is a gantry slave running with the master.

It is possible to combine master/booster operation with external position sensing. Up to 1 position sensor, 1 master and 2 boosters may be connected to one *LinMot*® servo controller. With this configuration the external position sensor must be connected to the first motor channel, the master to the second channel and the boosters to the other channels.

It is also possible to have external position sensors for gantry configurations. The sensors have to be connected directly before the motors. E.g. the first channel is master’s sensor, the second is the gantry master, the third is the slave’s sensor and the last motor is the gantry slave.

Position sensing

Position Sensor

With this parameter the user determines where the controller is to get its position information from. If external position sensing is employed, the resolution may be set in addition.

\Drives\Drive X\Advanced\Position Sensor	
Internal sensor 20µm	With this adjustment the actual position is determined with the inbuilt position sensing of the *LinMot®*. For strokes up to 1200mm, it is recommended to select this resolution. This is the default setting.
Internal sensor 40µm	With this adjustment the actual position is determined with the inbuilt position sensing of the *LinMot®* but the resolution is 40µm, so the motor can cover a stroke of up to 2400mm without shifting the home position.
External 20µm	With this adjustment the motor obtains its actual position from the position sensor connected on the channel above. If for example a motor is connected to channel B he then takes his actual position from the position sensor connected to channel A.
External 10µm	
External 5µm	
External 2.5µm	
External 1.25µm	
AB Enc1 1X	With this adjustment the motor obtains its actual position from the position sensor connected to an encoder link on the master encoder module (ME01-01/08 or ME01-02/08). The type of sensor is incremental AB, and the decode mode can be 1X (only rising edge of signal A), 2X (all rising edges of signal A and B) or 4X (any edge of signal A and B).
AB Enc1 2X	
AB Enc1 4X	
AB Enc2 1X	
AB Enc2 2X	
AB Enc2 1X	

In chapter “External position sensing” on page 53 it is explained how the adjusted resolution depends on the travel range.

Initializing (Homing)

Initialization

The position sensing employed in the motors of the *LinMot*® series allows relative position determination. When starting therefore, the reference position must be initialized just once. This is done by means of a so-called reference move.

The way a motor is initialized is defined by the parameters **Init Mode**, **Init Switches** and those in the **Init Config** directory.

Init Mode

The initialization mode lays down how the position zero of the slider is defined. There is a choice of the following modes:

\Drives\Drive X\Initialization\Init Mode	
Actual Position	The present position is defined as zero.
Auto Move Out	The slider moves out up to a mechanical stop. This position is set as zero.
Auto Move In	The slider moves in up to a mechanical stop. This position is set as zero.
Trig Move Out	The slider moves out till the zero is set by the positive slope of the trigger signal. If the trigger signal has already been set when initialization is started, the slider moves in till the trigger signal stops. Only then does the slider move out again, so that the zero can be fixed with the positive trigger slope.
Trig Move In	The slider moves in till the zero is set by the positive slope of the trigger signal. If the trigger signal has already been set when initialization is started, the slider moves out till the trigger signal drops. Only then does the slider move in again, so that the zero can be fixed with the positive trigger slope.

Init Switches

\Drives\Drive X\Initialization\Init Switches	
Init Once	If this parameter is activated, the motor is initialized in the **INIT** state for the first term only. If this parameter is not activated, the motors are initialized every time an active **INIT** signal is present.

Init Config

In the **Init Config** directory are the following initialization parameters:

\Drives\Drive X\Initialization\Init Config	
Init Speed	Defines the speed at which the slider moves when initializing.
Maximal Init Current	This parameter states the amperage that must be detected when initializing against a stop. It can thus be determined how firmly the linear motor is to press against the stop when initializing. The current is set as a percentage of the maximum current.
Home Position	After reaching the current set with the **Maximal Init Current** parameter when initializing, the motor is initialized. This position is now given the **Home Position** value.
Check Init Position	After defining the **Home Position** there is an automatic traverse to the position defined under **Check Init Position** and back to the **Home Position**. If the slider cannot reach the desired position (desired travel range not free), an error signal is given.
Initial Position	At the end of initialization the **Initial Position** is assumed. When the slider reaches this position, initialization is concluded.

In the example on the following page the meaning of these parameters is clarified.

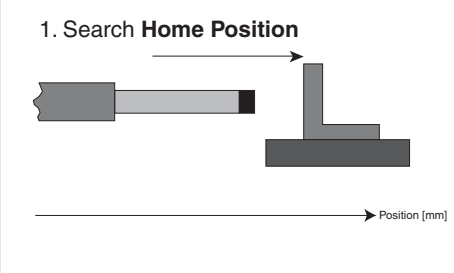
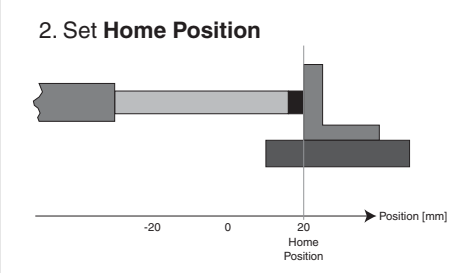
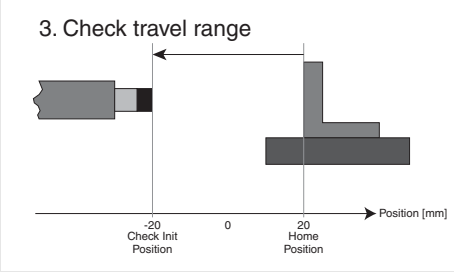
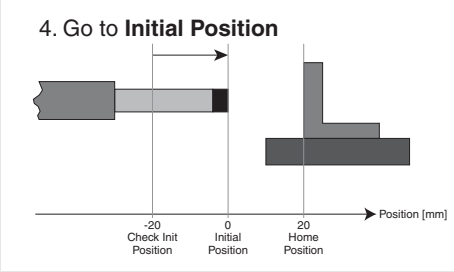
The example below shows the initialization procedure. The initialization parameters were set as follows.

Init Mode: Auto Move Out

Home Position: 20.0mm

Check Init Position: -20.0mm

Initial Position: 0.0mm

Action	Description
	The motor seeks its **Home Position**. Since **Auto Move Out** has been chosen as initialization mode, the slider will stroke out of the motor. The search is concluded as soon as the current reaches the **Maximal Init Current** value. This adjustment ensures that the motor presses firmly against its stop.
	After the motor has found the **Home Position** this position is given the **Home Position** value, in this example 20mm. With it the position axis is defined.
	After the **Home Position** has been defined, the **Check Init Position** is moved into. If an error occurs during this check traverse, the initialization is discontinued. If no check of the traverse is desired, the **Check Init Position** is set equal to the **Initial Position**.
	After checking the traverse range the **Initial Position** is assumed. After reaching it the initialization procedure is concluded and the motor is ready for operation.

Generating setpoints

In this directory the method of generating setpoints is adjusted.

Run Mode

The setpoint generating mode is defined with the **Run Mode** parameter. The following modes may be differentiated:

\Drives\Drive X\Set Value Generation\Run Mode	
Serial	The setpoint is given with a protocol via the serial interface.
Analog	The setpoint is given via the analog input corresponding to the motor. In the **Set Value Generation** directory the boundary setpoints for the input levels 0V and 10V is depicted linearly on the setpoint range defined by these boundary setpoints (maximum/minimum).
Continuous Curve	A motion profile stored in the servo controller is run through cyclically. The curve number is selected under **Curve Number** in the **Set Value Configuration** directory.
Trigger Curve	On the rising slope of the trigger signal a first, and on the falling slope a second motion profile stored in the servo controller are run through. The curve numbers are fixed in **Set Value Generation** directory under **Rise Curve Number** and **Fall Curve Number**. If the falling slope of the trigger signal ensues before the first profile is completed, the second profile is run immediately afterwards
Two Point	If the trigger signal corresponding to the drive is active, the value defined with the parameter **'1' Position** in the **Set Value Generation** is traversed. If the trigger signal is not active, the value is traversed that is defined with the parameter **'0' Position**.

Set Value Configuration

The **Set Value Configuration** directory contains all configuration parameters needed in connection with the setpoints.

\Drives\Drive X\Set Value Generation\Set Value Configuration	
Minimal Position L	Determines the minimal position for the motor (lower limit of positioning range).
Maximal Position L	Determines the maximal position for the motor (upper limit of positioning range).
0V Position L [1]	In the **Analog** mode this parameter defines this position to be assumed with 0V input voltage.
10V Position L [1]	In the **Analog** mode this parameter defines this position to be assumed with 10V input voltage.
'0' Position L [2]	Defines the position assumed in the **Two Point** mode when the trigger is at a logic 0.
'1' Position L [2]	Defines the position assumed in the **Two Point** mode when the trigger is at a logic 1.
Curve Number L [3]	In the **Continuous Curve** mode the motion profile stored in the servo controller and bearing the number in **Curve Number** is run over cyclically.
Rise Curve Number L [4]	In the **Trigger Curve** mode the motion profile with this number is run on the rising slope on the trigger signal.
Fall Curve Number L [4]	In the **Trigger Curve** mode the motion profile with this number is run on the falling slope on the trigger signal.
Curve Pos Offset L	This parameter sets the position offset of the motion profile.
Curve Amplitude L	This parameter sets the amplitude of the motion profile. The value range is from 0 to 100%.
Curve Speed L	This parameter sets the speed of the motion profile. The value range is from 0 to 100%.

1) Visible only in **Analog** mode
2) Visible only in **Two Point** mode
3) Visible only in **Continuous Curve** mode
4) Visible only in **Trigger Curve** mode

The diagram below shows the shaping of the analog input voltage to the position setpoint in the **Analog** mode.

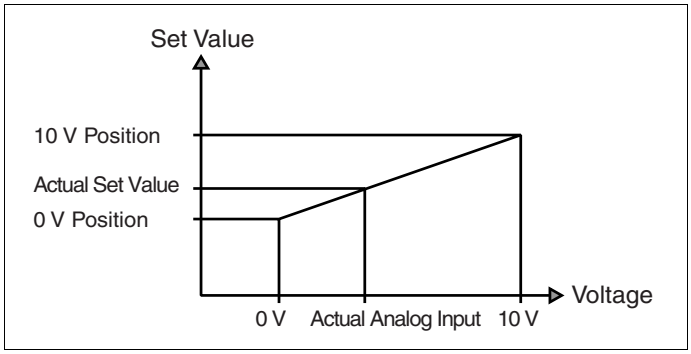


Figure 6-1: Set position with analog setpoint

The setpoint reached is the value interpolared linearly between the parameter **0V Position** and **10V Position**.

Filter Parameter

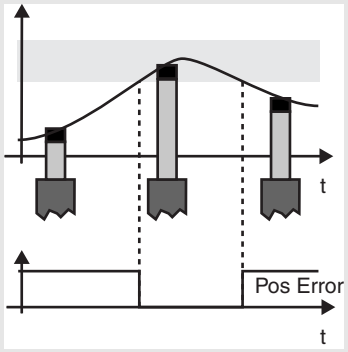
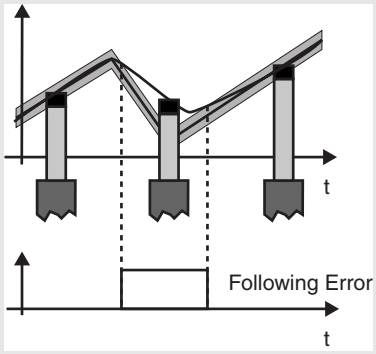
This directory includes all parameters needed for setpoint filtering.

\Drives\Drive X\Set Value Generation\Filter Parameter	
Max Speed L	This value sets the upper speed limit. It is **disregarded** when a motion profile is being run. The speed and acceleration are then taken from the motion profile.
Max Acceleration L	This value sets the upper limit for acceleration. It is **disregarded** when a motion profile is being run. The speed and acceleration are then taken from the motion profile.

Position monitoring

Position Monitoring

The firmware supports two kinds of position monitoring.

Position band monitoring	
For every motor there is a position band. If an active motor is outside its band, the digital output **POSITION ERROR OUT** becomes active.	
Following error monitoring	
The difference between setpoint and actual position must stay within certain limits. When this difference becomes excessive, if selected a warning or error is signalled. Reasons for this may be: setpoint motion profiles too fast, jumps in the setpoint motion profiles, excessive load mass, motor jammed.	

The following parameters serve to specify the limits:

\Drives\Drive X\Position Monitoring L	
Pos Range Min **Pos Range Max**	Define the upper and lower limits of the position band monitoring. If the actual position of the motor is below its limit, the digital output **POSITION ERROR OUT** is activated.
In Position- **In Position+**	These two parameters define how close to the desired target position the motor must be so that it is considered reached. If a motor is given a new set position (relative or absolute movement) or a motion profile is started, the **In Position** of the motor concerned goes to logic 0 till the slider of the motor is within a range around the setpoint.
Following Error- **Following Error+**	Define the maximum admissible following error. If the difference between setpoint and actual position is greater or smaller than the adjusted value, a warning or an error is signalled

Control parameters

Control Parameters

This directory includes the parameters needed for control. See also chapter 7. “Tips and Tricks for the controller”.

\Drives\Drive X\Control Parameters L	
Maximal Current	Determines the maximum current that can be applied to the motor by the controller. Through the maximum current this parameter also determines the maximum force the motor can provide.
Current Offset	Enables a static force sustained by the motor to be compensated.
P	Determines how the difference between setpoint and actual position is to be represented by the demand current. A setting of 1 A/mm causes a current of 1 ampere for a position deviation of 1 mm.
D	Determines how the difference between setpoint and actual speed is to be represented by the current setpoint. An adjustment of 4 A/(m/s) causes a current of 4 amperes for a speed difference of 1 mm/s.
I	Determines how the time integral of the position deviation is to be represented by the current setpoint. An adjustment of 100 A/(mm*s) causes a current of 1 ampere for a position difference of 0.1 mm over a time of 0.1 s

\Drives\Drive X\Control Parameters L	
FF Friction	Determines what constant current is to be fed forward for a slider motion. The sign for the feed-forward current depends on the direction of the motor stroke. This parameter may be used to compensate any friction present.
FF Acceleration	Determines what current must be fed forward to obtain the desired acceleration. A value of 100mA/(m/s2) causes a current of 100mA to be fed forward for an acceleration difference of 1m/s^2.
FF Deceleration	Determines what current must be fed forward to obtain the desired deceleration. A value of -100mA/(m/s^2) causes a current of -100mA to be fed forward for an deceleration difference of 1m/s^2.
Noise Dead Band	This parameter defines the width of the noise filter dead band. This feature is to reduce the noise when the motor stands still. Because this filter will reduce the accuracy in positioning, it should only be turned on if the acoustic noise is really disturbing and, when turned on, the value should be set to the minimum where the noise disappears. The noise dead band filter becomes active, when the demand position doesn't change anymore. When the integral position control parameter is set to zero, the filter will freeze the motor current unless the actual motor position deviates more than the noise dead band width is defined from the average of the last eight actual motor positions before this filter becomes active. When the integral position control parameter is turned on, the filter will freeze the motor current unless the actual motor position deviates more than the noise dead band width is defined from the demand position.

Control modes

Control Switches

This parameter is visible only with a servo controller of the E100 series and determines whether maximum current is to be 2 or 3 amperes. The maximum current may be fine tuned with the **Maximal Current** parameter in the **Control Parameters** directory.

\Drives\Drive X\Control Switches	
Current ( ) 2A (x) 3A [1]	If this parameter is selected the maximal current is limited to 3 otherwise to 2 amperes.

1) Only visible for the series E100 controller units

Commutation

Commutation

In this directory are the parameters determining the commutation of the connected drives are to be powered. On a linear motor the commutation affects the following operating variables:

- force ripple
- heat losses in the motor
- motor dynamics

The following commutation modes may be selected:

\Drives\Drive X\Advanced\Commutation		
Sine (Default)	Force ripple: heat losses in the motor: motor dynamics: In most cases it is advisable to select this standard commutation adjustment.	little small medium
Trapezoid	Force ripple: Heat losses in motor: Motor dynamics:	large large medium to high
Block	Force ripple: Heat losses in motor: Motor dynamics: Because of the very large heat losses, this commutation may be chosen only where highest dynamics in association with **low duty cycle** are demanded. The duty cycle is the ratio between the moving time and standstill time of the motor.	very large very large!!! high

The parameters for the maximal current are in the directory **\Drives\Drive X\ Control Parameters**.

Error handling

Handling errors on the motor side is described in this subsection. General information on errors and warnings may be obtained from the chapter “Error handling” on page 62.

Error Mask
Warn Mask

The directories below show for the motors which internal errors lead to the error state (**Error Mask**), which errors generate a warning (**Warn Mask**) and what is to happen in the event of an emergency stop (**Emergency Stop**).

The user cannot access all error and warn parameters because the hardware might suffer damage through incorrect settings.

\Drives\Drive X\Error Handling\Error Mask	
Drive Type Mismatch R	The connected motor type does not match the one selected or the connected motor is defective.
Curve Error R	A setpoint motion profile cannot be found.
Slider Missing R	The slider is missing or the motor has not been connected properly.
Init Failed R	An error has occurred while initializing the motor.
Drive Following Error	A motor has a following error.
Drive Too Hot Calculated	A motor was temporary overloaded. Possible causes are: the motor was blocked or overloaded (load mass too big, trajectory too fast, ...). If this parameter is unselected, it can be possible that an overheating of the motor, which is caused ba a shorttime overload, cannot be detected anymore. The motor may be damaged.
Drive Too Hot Sensor R	An overheated motor has been detected. It has been overloaded and/or cooled inadequately.

Most parameters in the **Warn Mask** directory can be edited, unlike those in the **Error Mask** directory. Only the warning **Drive Init Not Done** cannot be cleared. This warning is maintained till a motor has been initialized and prevents the servo controller assuming the **RUN** state as long as a motor is not initialized.

\Drives\Drive X\Error Handling\Warn Mask	
Slider Missing	Slider missing or motor incorrectly connected.
Drive Init Not Done R	A motor has been connected incorrectly or not at all.
Drive Following Error	The motor has a following error.
Pos Range Indicator	The slider is outside the position range defined by **Pos Range Min/Max**. If this warning is activated, make sure that the **Pos Range** parameters are adjusted so that the signal is no longer present after initializing, otherwise it will be possible to ascertain only with great difficulty whether all motors have been initialized properly. The output will only be activated with an AT servo controller if the parameter **Pos Error Output** in the directory **\System\IO Configuration** is activated. With an MT servo controller in the MT mode (parameter **MT** in directory **\System\Command Interface** activated) the output is only activated if the parameter **Pos Range** in directory **\Multi Trigger\ Output Configuration\Output 3** is activated.
Drive Too Hot Calculated	A motor was temporary overloaded. Possible causes are: the motor was blocked or overloaded (load mass too big, trajectory too fast, ...). If this parameter is unselected, it can be possible that an overheating of the motor, which is caused ba a shorttime overload, cannot be detected anymore. The motor may be damaged.
Drive Hot Sensor	Excessive motor heating has been detected. Motor overloaded and/or inadequately cooled.
In Position	A motor has reached the demand position. The corresponding output **Msg Output** will be activated if the parameter **Msg Output** in the directory **\System\IO Configuration** is activated and the parameter **In Position** in the directory **\System\Error Handling\Msg Mask** is activated.

Emergency Stop Mode

The behavior of the motor after an emergency stop can be defined with the following modes:

\Drives\Drive X\Error Handling\Emergency Stop\Emergency Stop Mode	
Off	Motor no longer controlled. Position still read in. In this mode the phases no longer have current.
Freeze	The motor decelerates[1] and stays in that position. The motor remains in operation.
Goto Position	The motor goes[1] to the **Emergency Stop Position** and stays there. The motor remains in operation.

1) The speed and acceleration value of these movements can be set with the parameters **Maximal Speed** and **Maximal Acceleration** in the directory **EmergencyConfiguration**.

Emergency Configuration

This directory contains only one parameter defining the position to which the motor is moved in the event of an emergency stop.

\Drives\Drive X\Error Handling\Emergency Stop\Emergency Configuration [1]	
Stop Position L [2]	If the **Drive Goto Position** emergency stop mode is selected, when an emergency stop occurs the position defined under **Emergency Stop Position** will be assumed at once. The motor remains in operation.
Max Speed L	Sets the speed with which the motor goes to the **Stop Position**.
Max Acceleration L	Sets the acceleration with which the motor accelerates/decelerates if the **STOP** signal is activated.

1) Visible only if the parameter **Emergency Stop Mode** is set to **Goto Position** or **Freeze**.
2) Visible only if the parameter **Emergency Stop Mode** is set to **Goto Position**.

6.5 Stepper motor parameters

Any axis on the servo controller can be configured to drive a stepper motor directly without the need for a translator. This is useful in applications where linear and rotary motion are needed.

In this section all parameters needed for configuration of stepper motors are explained. To make them visible the parameter **Stepper** must be selected in the **\Drives\Drive X\Type** directory.

Master/Booster Operation

Master/Booster

In the stepper motor mode this parameter must always be set to the **Master** value. **Master/Booster** configurations are not supported in the stepper motor mode.

\Drives\Drive X\Master/Booster	
Master	Must be selected if a stepper motor is controlled.

Initializing

Initialization

Normal stepper motors allow only a relative determination of position. When starting therefore, the reference position must be initialized just once. This is a so-called reference move.

The initialization mode of a stepper motor can be defined using the parameters **Init Mode**, **Init Switches** and the parameters in the **Init Config** directory.

Init Mode

The initialization mode establishes how the position zero of the stepper motor is defined. There is a choice of the following modes:

\Drives\Drive X\Initialization\Init Mode	
Actual Position	The momentary actual position is defined as zero.
Trig Turn Left	The stepper motor turns counter clockwise till the zero is fixed by the positive slope of the trigger signal. If the trigger signal is already active at the start of initialization, the rotor turns in the opposite direction till the trigger signal drops. Only then does it turn left again so that the zero can be fixed with the positive trigger slope.
Trig Turn Right	The stepper motor turns clockwise till the zero is fixed by the positive slope of the trigger signal. If the trigger signal is already active at the start of initialization, the rotor turns in the opposite direction till the trigger signal drops. Only then does it turn right again so that the zero can be fixed with the positive trigger slope.

Init Switches

\Drives\Drive X\Initialization\Init Switches	
Init Once	If this parameter is activated the motor is initialized only the first time in the **DRIVE INIT** state. If it is not activated, the motors will be initialized every time when changing from the **DISABLE** to the **DRIVE INIT** state.

Init Config

In the **Init Config** directory are the following initialization parameters:

\Drives\Drive X\Initialization\Init Config	
Init Speed	Defines the speed at which motor rotates when initializing.
Home Position	When the reference move is completed the actual position is assigned to this value.
Initial Position	At the end of initialization the **Initial Position** is moved into. Initialization is concluded when the motor reaches this position.

Generating Setpoints

Run Mode

The setpoint generating mode is defined with the **Run Mode** parameters. The following modes may be distinguished:

\Drives\Drive X\Set Value Generation\Run Mode	
Serial	The setpoint is given with a protocol via the serial interface.
Analog	The setpoint is given via the analog input corresponding to the motor. In the **Set Value Generation** directory the boundary setpoints may be fixed for the 0V and 10V input levels. The analog input range between 0V and 10V is formed linearly on the setpoint range defined by these boundary setpoints (maximum/minimum).
Continuous Curve	A setpoint motion profile stored in the servo controller is run through cyclically. The motion profile numbers are selected under **Curve Number** in the **Set Value Configuration** directory.
Trigger Curve	On the rising slope of the trigger signal a first, and on the falling slope a second motion profile stored in the servo controller are run through. The curve numbers are fixed in **Set Value Generation** directory under **Rise Curve Number** and **Fall Curve Number**. If the falling slope of the trigger signal ensues before the first profile is completed, the second profile is run immediately afterwards
Two Point	If the trigger signal corresponding to the drive is active, the value defined with the parameter **'1' Position** in the **Set Value Generation** is traversed. If the trigger signal is not active, the value is traversed that is defined with the parameter **'0' Position**.

Set Value Configuration

The **Set Value Configuration** directory contains all configuration parameters needed in connection with the setpoints.

\Drives\Drive X\Set Value Generation\Set Value Configuration	
Minimal Position L	Determines the minimal position for the motor (lower limit of positioning range).
Maximal Position L	Determines the maximal position for the motor (upper limit of positioning range).
0V Position L [1]	In the **Analog** mode this parameter defines the position to be assumed with 0V input voltage.
10V Position L [1]	In the **Analog** mode this parameter defines the position to be assumed with 10V input voltage.
'0' Position L [2]	Defines the position assumed in the **Two Point** mode when the trigger is at a logic 0.
'1' Position L [2]	Defines the position assumed in the **Two Point** mode when the trigger is at a logic 1.
Curve Number L [3]	In the **Continuous Curve** mode the motion profile stored in the servo controller and baring the number in **Curve Number** is run over cyclically
Rise Curve Number L [4]	In the **Trigger Curve** mode the motion profile with this number is run on the rising slope of the trigger signal.
Fall Curve Number L [4]	In the **Trigger Curve** mode the motion profile with this number is run on the falling slope of the trigger signal.
Curve Pos Offset L	This parameter sets the position offset of the motion profile.
Curve Amplitude L	This parameter sets the amplitude of the motion profile. The value range is from 0 to 100%.
Curve Speed L	This parameter sets the speed of the motion profile. The value range is from 0 to 100%.

1) Visible only in **Analog** mode
2) Visible only in **Two Point** mode
3) Visible only in **Continuous Curve** mode
4) Visible only in **Trigger Curve** mode

Figure 6-1, "Set position with analog setpoint", on page 78 shows how the setpoint behaves in the **Analog** mode.

Filter Parameters

This directory includes all parameters needed for setpoint filtering.

\Drives\Drive X\Set Value Generation\Filter Parameter	
Max Speed L	This value sets the upper speed limit. It is disregarded when a motion profile is being run. The speed and acceleration are then taken from the motion profile.
Max Acceleration L	This value sets the upper limit for acceleration. It is disregarded when a motion profile is being run. The speed and acceleration are then taken from the profile.

Position Monitoring

The firmware supports two kinds of position monitoring that are explained in chapter “Position monitoring” on page 79. Because the stepper motor has no position feedback it is not possible to use the following error monitoring.

The following parameters serve to specify the limits:

\Drives\Drive X\Position Monitoring L	
Pos Range Min	Defines the upper and lower limits of the position band monitoring. If the actual position of drive is below its limit, the digital output **POSITION ERROR OUT** is activated.
Pos Range Max	
In Position -	These two parameters define how close to desired target position the motors must be so that this is considered reached. Active only in the MT mode. If a motor is given a new set position (relative or absolute movement) or a motion profile is started, the **In Position** of the motor concerned goes to logic 0 till the slider of the motor is within a range around the setpoint.
In Position +	

Control Parameters

This parameter sets the current for the stepper motor.

\Drives\Drive X\Control Parameters	
Maximal Current	This parameter sets the current for the stepper motor. If the stepper motor is not moving the current is reduced to 50%.

Control Modes

This parameter is visible only with a servo controller of the E100 series and determines whether maximum current is to be 2 or 3 amperes. The maximum current may be fine tuned with the **Maximal Current** parameter in the control directory.

\Drives\Drive X\Control Switches	
Current ( ) 2A (x) 3A [1]	With this parameter a maximum current of 3 A is elected, otherwise it is 2 A.

1) Visible only with a servo controller of the E100 series

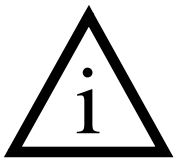
Commutation

In this directory are the parameter determining the commutation of the connected drives. The commutation decides how the two phases of the connected motors are to be powered. With the stepper motors the commutation affects the following operating variables:

- resolution
- smooth running
- maximum attainable speed

\Drives\Drive X\Advanced\Commutation		
Auto (Default)	Resolution: Max. speed:	4 quarter-steps / step high
	With this commutation there is automatic shuttling to and fro between the commutations explained below, depending on the speed. The advantage is the high maximum speed combined with good resolution.	
Micro Step	Resolution: Max. speed:	4 quarter-steps / step low
	The advantage of this commutation is the attainable resolution and low-vibration running. The low maximum speed attainable is a disadvantage.	
Half Step	Resolution: Max. speed:	2 quarter-steps / step medium
Full Step	Resolution: Max. speed:	1 quarter-steps / step high
	The advantage of this commutation is the maximum speed attainable. on the other hand the resolution is low.	

Current resolution



Because the current drivers in the servo controller have only limited resolution, only the combinations of maximum current (phase current) and commutation set out below should be used.

Commutation	Maximum Current			
	LinMot E100/E200/E400[1]		LinMot E1000/ E2000/ E4000	LinMot E1001/ E2001/ E4001
	2A	3A		
Auto/Micro Step	1 / 2A	1.5 / 3A	0.75 / 1.5 / … / 6A	0.75 / 1.5 / … / 7.5A
Half Step	0.5 / 1 / 1.5 / 2A	0.75 / 1.5 / 2.25 / 3A	0.375 / 0.75 / … / 6A	0.375 / 0.75 / … / 7.875A
Full Step	> 0.125A	> 0.178A	> 0.05A	> 0.05A

Table 6-4: Maximum current adjustment for stepper motors

1) For all E100/E200/E400 device which have a serial number bigger than ----.001.200 only the 3A column is valid.

Current Reduction

If the rotor of the stepper motor performs no movement, the phase current is halved. This reduction ensues automatically 200 ms after standstill. The parameter for the motor current can be found in the directory **Drive\Drive X\Control Parameters**.

Error Handling

Handling errors on the motors are described in this subsection. General information on errors and warnings may be obtained from the chapter “Error handling” on page 62.

Error Mask
Warn Mask

The directories below shows which internal errors lead to the error state (**Error Mask**), which errors generate a warning (**Warn Mask**) and what is to happen in the event of an emergency stop (**Emergency Stop**).

The user cannot access all error and warn parameters because the hardware might suffer damage through incorrect adjustments.

\Drives\Drive X\Error Handling\Error Mask	
Drive Type Mismatch R	The connected motor type does not match the one selected or the connected motor is defective.
Curve Error R	A motion profile cannot be found.

The **Drive Init Not Done** parameter cannot be deselected, so that the servo controller cannot assume the **RUN** state if a motor has not been initialized yet.

\Drives\Drive X\Error Handling\Warn Mask	
Drive Init Not Done R	A motor has been connected incorrectly or not at all.
Pos Range Indicator	The slider is outside the position range defined by **Pos Range Min/Max**. If this warning is activated, make sure that the **Pos Range** parameters are adjusted so that the signal is no longer present after initializing, otherwise it will be possible to ascertain only with great difficulty whether all motors have been initialized properly. The output will only be activated with an AT servo controller if the parameter **Pos Error Output** in the directory **\System\IO Configuration** is activated. With an MT servo controller in the MT mode (parameter **MT** in directory **\System\Command Interface** activated) the output is only activated if the parameter **Pos Range** in directory **\Multi Trigger\ Output Configuration\Output 3** is activated.
In Position	A motor has reached the demand position. The corresponding output **Msg Output** will be activated if the parameter **Msg Output** in the directory **\System\IO Configuration** is activated and the parameter **In Position** in the directory **\System\Error Handling\Msg Mask** is activated.

Emergency Stop Mode

The behavior of the motor after an emergency stop can be defined with the following modes:

\Drives\Drive X\Error Handling\Emergency Stop\Emergency Stop Mode	
Off	Motor no longer controlled. Position still read in. In this mode the phases no longer have current.
Freeze	The motor decelerates[1] and stays in that position. The motor remains in operation.
Goto Position	The motor goes[1] to the **Emergency Stop Position** and stays there. The motor remains in operation.

1) The speed and acceleration value of these movements can be set with the parameters **Max Speed** and **Max Acceleration** in the directory **Emergency Configuration**.

Emergency Configuration

This directory contains the parameters defining with what parameters the motor is moved if an emergency stop is activated with the **STOP** signal.

\Drives\Drive X\Error Handling\Emergency Stop\Emergency Configuration [1]	
Stop Position L [2]	If the **Drive Goto Position** emergency stop mode is selected, when an emergency stop occurs the position defined under **Emergency Stop Position** will be assumed at once with ultimate regulation to this. The motor remains in operation.
Max Speed L	Sets the speed with which the motor goes to the **Stop Position**.
Max Acceleration L	Sets the acceleration with which the motor accelerates/decelerates if the **STOP** signal is activated.

1) Visible only if the parameter **Emergency Stop Mode** is set to **Goto Position** or **Freeze**.
2) Visible only if the parameter **Emergency Stop Mode** is set to **Goto Position**.

6.6 Solenoid parameters

Any axis on the servo controller can be configured to control a solenoid.

In this section all parameters needed for configuration of solenoids are explained. To make them visible in the **Parameter Inspector**, the **Drive Type** parameter must be selected in the **\Drives\Drive X\Type** directory.

Master / Booster

In the solenoid mode this parameter must be always be set to **Master**.

\Drives\Drive X\Advanced\Master / Booster	
Master	The connected actuator is a master.

Generating setpoints

In this directory the setpoint generation method is adjusted.

Run Mode

The setpoint generating mode is defined with the **Run Mode** parameter. The following modes may be distinguished:

\Drives\Drive X\Set Value Generation\Run Mode	
Serial	The setpoint is given with a protocol via the serial interface. The customer-specific protocol is provided in the customized application software.
Analog	The setpoint is given via the analog input corresponding to the motor. In the **Set Value Generation** directory the boundary setpoints may be fixed for the 0V and 10V is formed linearly on the setpoints range defined by these boundary setpoints (maximum/minimum).
Continuous Curve	A setpoint motion profile stored in the servo controller is run through cyclically. The curve numbers are selected under **Curve Number** in the **Set Value Configuration** directory.
Trigger Curve	On the rising slope of the trigger signal a first, and on the falling slope a second motion profile stored in the servo controller are run. The curve numbers are fixed in the **Set Value Generation** under **Rise Curve Number** and **Fall Curve Number**. If the falling slope of the trigger signal ensues before the first profile is completed, the second curve is run immediately afterwards.
Two Point	If the trigger signal corresponding to the drive is high, the value defined with the parameter **'1' Current** is driven. If the trigger signal is low, the value defined with the parameter **'0' Current** is driven.

Set Value Configuration

The **Set Value Configuration** directory contains all configuration parameters needed in connection with the setpoints.

\Drives\Drive X\Set Value Generation\Set Value Configuration	
Minimal Current L	Determines the minimal current for the solenoid (lower limit of current range).
Maximal Current L	Determines the maximal current for the solenoid (upper limit of current range).
0V Current L 👁[1]	In the **Analog** mode this parameter defines the current which is to flow through the winding with 0V input voltage with solenoid control.
10V Current L 👁[1]	In the **Analog** mode this parameter defines the current which is to flow through the winding with 10V input voltage with solenoid control.
'0' Current L 👁[2]	With solenoids this value defines the current driven with a logic 0 of the trigger.
'1' Current L 👁[2]	With solenoids this value defines the current driven with a logic 1 of the trigger.
Curve Number L 👁[3]	In the **Continuous Curve** mode the motion profile stored in the servo controller and bearing the number in **Curve Number** is run over cyclically.
Rise Curve Number L 👁[4]	In the **Trigger Curve** mode the motion profile with this number is run on the rising slope of the trigger signal.
Fall Curve Number L 👁[4]	In the **Trigger Curve** mode the motion profile with this number is run on the falling slope of the trigger signal.
Curve Current Offset L	This parameter sets the current offset of the motion profile.
Curve Amplitude L	This parameter sets the amplitude of the motion profile. The value range is from 0 to 100%.
Curve Speed L	This parameter sets the speed of the motion profile. The value range is from 0 to 100%.

1) Visible only in **Analog mode**
2) Visible only in **Two Point mode**
3) Visible only in **Continuous Curve mode**
4) Visible only in **Trigger Curve mode**

The diagram below shows the shaping of the **analog** input voltage to the current setpoint in the **Analog** mode.

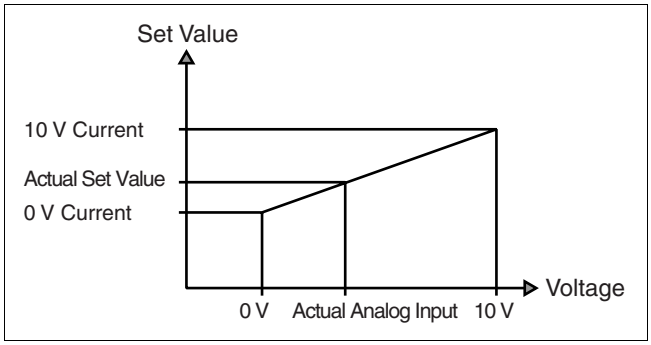


Figure 6-2: Set current with analog setpoint

The setpoint reached is the value interpolated linearly between the parameter **0V Current** and **10V Current**.

Control Parameters

This directory includes the parameters needed for control and regulation. See also page 82 for tips and tricks.

\Drives\Drive X\Control Parameters	
Maximal Current L	Determines the maximum current and therefore the maximum force.

Control Modes

This parameter is visible only with a servo controller of the E100 series and determines whether maximum current is to be 2 or 3 amperes. The maximum current may be narrowed still closer with the **Maximal Current** parameter in the **Control Parameters** directory.

\Drives\Drive X\Control Switches	
Current ( ) 2A (x) 3A [1]	With this parameter a maximum current of 3 A is selected, otherwise it is 2 A.

1) Visible only with a servo controller of the E100 series.

Error Handling

Handling errors on the motors are described in this subsection. General information on errors and warnings may be obtained from chapter “Error handling” on page 62.

Error Mask

The directory below shows which internal errors lead to the error state (**Error Mask**) and what is to happen in the event of an emergency stop (**Emergency Stop**).

\Drives\Drive X\Error Handling\Error Mask	
Drive Type Mismatch R	The connected motor type does not match the one selected or the connected motor is defective.
Curve Error R	A motion profile cannot be found.

Emergency Stop Mode

The behavior of the solenoid after an emergency stop can be defined with the following mode:

\Drives\Drive X\Error Handling\Emergency Stop\Emergency Stop Mode	
Off	Solenoid no longer has current.
Freeze	The actual amperage at the moment of the emergency stop is frozen. The solenoid remains in operation.
Set Current	At any emergency stop the current defined under **Stop Current** is driven at once.

Emergency Configuration

This directory contains only one parameter defining the current which is driven into the solenoid in the event of an emergency stop.

\Drives\Drive X\Error Handling\Emergency Stop\Emergency Configuration	
Stop Current L [1]	If the **Set Current** emergency stop mode is selected, when an emergency stop occurs this current is driven at once into the connected solenoid.

1) Visible only if the **Emergency Stop Mode** is set to **Set Current**.

6.7 Position sensing parameters

With the external sensing it is possible to enhance the repeat accuracy and the linearity of the *LinMot*® motors. Sine/cosine encoders with 1 to 5 mm pole distance are supported. The external sensing can be configured with two parameters. In chapter “External position sensing” on page 53 this subject is explained in more detail.

Sensor Period

With this parameter the pole distance of the connected encoder is set. Either 1 mm or 5 mm may be selected.

\Drives\Drive X\Sensor Configuration\Sensor Period	
1 mm	Must be chosen if the connected sensors have 1 mm pole distance.
2 mm	Must be chosen if the connected sensors have 2 mm pole distance.
5 mm	Must be chosen if the connected sensors have 5 mm pole distance.

Sensor Direction

With this parameter the direction of the sensor in relation to the motor is set.

\Drives\Drive X\Sensor Configuration\Sensor Direction	
Positive	Must be set if the sine signal is to come before the cosine when the slider strokes out of the motor.
Negative	Must be set if the cosine signal is to come before the sine when the slider strokes out of the motor.

Error Mask
Warn Mask

The directory below shows which internal errors lead to the error state (**Error Mask**) and which warnings (**Warn Mask**) are available.

\Drives\Drive X\Error Handling\Error Mask	
Drive Type Mismatch R	The connected motor type does not match the one selected or the connected motor is defective.
Slider Missing R	The sensor is missing or incorrectly mounted or connected.

\Drives\Drive X\Error Handling\Warn Mask	
Slider Missing	The sensor is incorrectly mounted or connected.

6.8 MT parameters

The parameters for the MT servo controller serve to configure the inputs and outputs. They are available only on the MT servo controllers (e.g. E400-MT).

Jitter

Jitter Filter

Via the four digital inputs up to 16 states can be defined, which may be recalled individually by the higher-level control. When more than one digital input signal changes its state at the same time, jitter effects may occur.

If for example state 3 (0011) is called from state 0 (input combination 0000), the last two input bits ought to change at exactly the same moment. If this does not happen, during the change the status 1 (0001) or 2 (0010) will be assumed briefly. If the servo controller were now to detect these transient states, undesirable consequences would result. To prevent this, a new state is assumed only after the input signals have remained stable for an adjustable time interval.

Multi Trigger\Jitter Filter	
Time	Time interval during which the input signals must be stable so that a change of state is performed.

Acknowledge

Acknowledge

This parameter defines how long the signal **In Position** at least stays at zero after a command has been executed. The signal **In Position** goes to one if this time has expired **and** the motor has reached its wanted position.

Multi Trigger\Acknowledge	
Time	Minimal time during the signal **In Position** goes to zero after a command has been executed.

Output Signals

Output Configuration

Outputs 3 and 4 can be configured by the user by means of the parameters in the directory **Multi Trigger\Output Configuration**.

Output 3
Output 4

The function of output 3 and 4 can be selected with this parameter.

Multi Trigger\Output Configuration\Output 3	
None	Output not driven
In Pos A / In Pos B **In Pos C / In Pos D**	**In Position** signals from motor A, B, C and D
In Pos A*B	AND gating of relevant **In Position** signals
In Pos C*D	
In Pos A*B*C*D	
Pos Range	**Pos Range** signal is given

Multi Trigger\Output Configuration\Output 4	
None	Output not driven
In Pos A / In Pos B **In Pos C / In Pos D**	**In Position** signal from motors A, B, C and D
In Pos A*B	AND gating of relevant **In Position** signals
In Pos C*D	
In Pos A*B*C*D	

6.9 PROFIBUS Parameters

In this section the parameters of the PROFIBUS-DP servo controller are explained.

Diagnosis Priority

With the **Diagnosis Priority** parameter the user decides the priority with which the *LinMot®* servo controller request diagnosis from the PLC in the event of an error.

PROFIBUS-DP\Diagnose Priority	
None	No diagnose requested from the PLC.
Low	Diagnosis requested from PLC with low priority. The cyclic program (OB1) of the PLC is interrupted by a low-priority OB.
High	Diagnosis requested from PLC with high priority. The cyclic program (OB1) of the PLC is interrupted by a high priority OB.

Byte Order Datamodules

The **Byte Order Datamodules** parameter determines the byte order with which the data modules are to be evaluated and dispatched on the *LinMot®* servo controller.

PROFIBUS-DP\Byte Order Datamodules	
Reversed	When this parameter is selected, the byte order of the data modules is reversed. With Siemens PLC control systems this parameter should be selected.

Info

The parameters in the **Info** directory are valid only when the servo controller has been started.

PROFIBUS-DP\Info	
Slave Node Address	Shows the PROFIBUS address of the *LinMot®* servo controller.
Master Node Address	Shows the PROFIBUS address of the master.
Baudrate	Shows the identified baud rate. This parameter is only valid if the servo controller is properly recognized by the DP master and is in the data exchange mode.
Bus Cycle Time	Shows the PROFIBUS cycle time which the controller has actually measured.

Interface Card Type

With the selector parameter **Interface Card Type** the type of interface card can be selected.

Interface Card Type	
None	No interface card is attached or used.
DI01-08/08	A digital I/O module DI01-08/08 is attached, with 8 digital inputs and 8 digital outputs.
ME01-01/08	A master encoder module ME01-01/08 is attached, which has one encoder link, 8 digital inputs and 8 digital outputs.
ME01-02/08	A master encoder module ME01-02/08 is attached, which has two encoder links, 8 digital inputs and 8 digital outputs.

7. Tips and Tricks for the controller

7.1 Introduction

This section enlarges upon tips and tricks for the new controller in textual form, without going into the complicated theory. It cannot and does not intend to impart theoretical knowledge for adjusting the controller. The interested user who likes to wrestle with theory is advised to consult the relevant literature.

The following picture shows the structure of the controller. The words printed with italic letters announce parameters which can be adjusted. Basically it is a PID-Controller with Feed-Forward structures and an additional v/a-Limiter for the prefiltering of the reference signal. As an option there is a profile generator integrated.

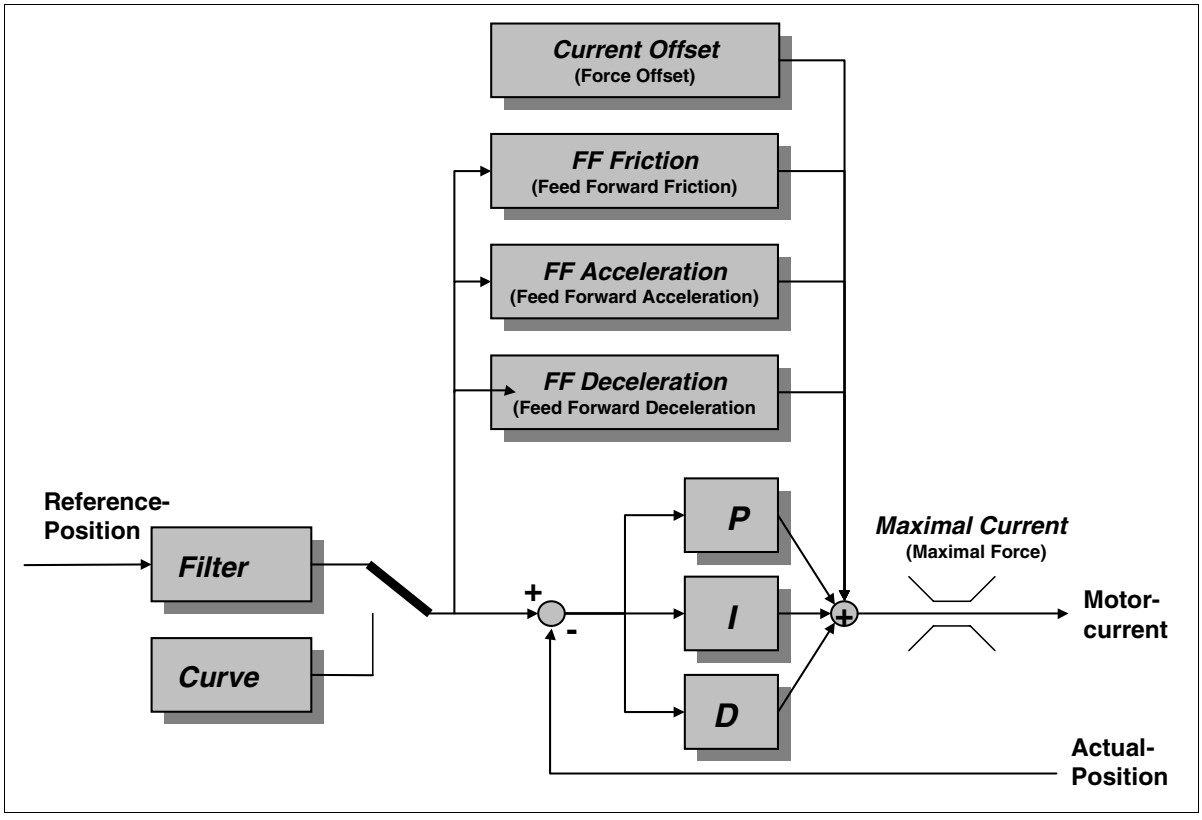


Figure 7-1: Structure of the controller

The default setting of the controller parameters is suitable for operating the motors under a lot of normal conditions. Specially if the load mass exceeds that of the slider by factors, or when using motors with long sliders, the controller should be adapted to this duty. Adapting the control parameters is advisable also if major friction forces occur or highly dynamic movements are demanded from the motor.

It is important to follow exactly the following guide lines. Tuning without following proper rules is nearly impossible or needs a lot of time.

7.2 Selection between PD- or PID- Controller

PD-Controller

A PD controller is one working with only proportional and derivative action (I=0). This type of controller is simple to adjust and has a very robust behaviour. Its disadvantage is that any static deviations present are not corrected automatically

PID Controller

A PID controller works with proportional, integral and derivative control. This kind of controller corrects automatically any static controller deviations by virtue of its I action. The disadvantage of this controller is that system oscillations may occur. When adjusting the controller the values may be taken over from the PD controller, and the I term increased slowly. The higher the I term is set, the faster the controller will correct any position deviations occurring. An overlarge I term may lead to instabilities in the control behavior however. With high load masses a small I term is advisable thereof

7.3 Adjusting of the prefilter (Filter)

The prefilter limits the max acceleration and velocity to the goal of the user. Using the prefilter it is possible that a PC or PLC can send a rectangular position jump but the motor moves smoothly, limited by the max acceleration and velocity. Basically the reference signal is not allowed to change 'faster' than the motor can follow! The right set up of the prefilter is more important than the optimal tuning of the controller!

Which max velocity and acceleration a motor can reach depends in a complex way with the parameters of the application (mass, friction, profile, amplifier, ...). It is proposed that the user simulates the application with the program *LinMot*® Designer (see LinMot CD or www.linmot.com) to calculate the possible max values. In the case of using profiles the prefilter is switched off. Max values for acceleration and velocity must be observed during the creation of the profiles.

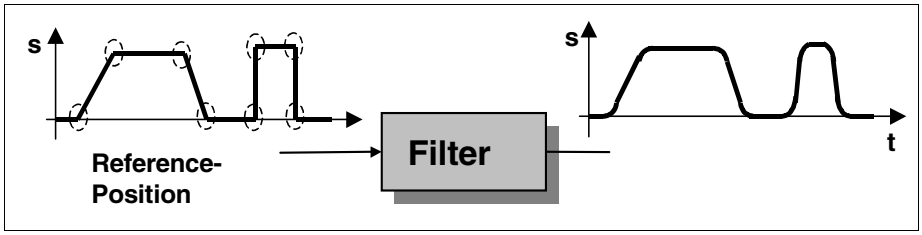


Figure 7-2: Reference signal before and after Filter

7.4 Using profiles for reference position

Using a mode which is based on profiles it is important to realize the following 4 points:

a) the motor must be able to fulfill the profile based on the maximal possible velocity and accelerations. The profile should not change faster than the motor is able to follow. It is proposed that the user simulates the application with the program *LinMot®* Designer (see LinMot CD or www.linmot.com) to calculate the possible max values.

b) All profiles should be smoothed. No jump in position or velocity is allowed. Use sine or hyperbolic functions while generating ramps which are offered by the Curve Inspector of *LinMot®* Talk.

c) Accept the number of points which are proposed by the “Curve Inspectors’ of *LinMot®* Talk (use Release 1.3.9 or higher). If the movement sounds ‘raw and hard’ reduce (!) the number of points!

d) If the profile will be produced by a third party program the following rules should be used: every profile should consist at least of 16 points but the time between the points should not be shorter than 1 ms (if there is a position step of 20 mm in 14 ms the number of points should be 14). In any other cases the distance between the points should be about 5 ms.

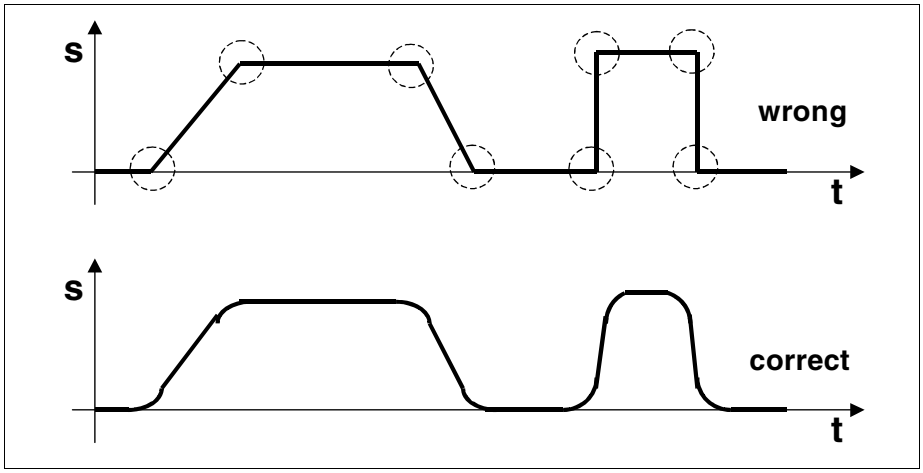


Figure 7-3: Profiles must be ‘smoothed’

7.5 Adjustment of the Feed-Forward Parameters

The term **Feed Forward** is used in control engineering to imply the anticipatory calculation of a control variable. This **anticipation** enables the controller to respond much better to the problem faced. When the controller **knows** that there is high friction in the system and knows the coefficient of friction, it can provide in advance the current necessary for a desired forward motion in order to overcome this friction. There is then much less discrepancy from the outset between the actual and target positions.

The controller integrated in the *LinMot®* servo controller includes these three anticipatory parameters with which the control behavior and hence the attainable dynamics can be improved

FF Friction

With the **FF Friction** parameter the sliding friction of the system can be compensated. Its value may be calculated as follows:

FF Friction = F_{FR} / c_f	FF Friction	Feed forward friction [A]
	F_{FR}	Sliding friction [N]
	c_f	Force constant motor [N/A]

where F_{FR} is the sliding friction and c_f is the force constant of the chosen motor.

Tip:
FFR can be measured with a spring scale (disconnect motor) and the value for the force constant cf can be read from the data sheets. In applications with long strokes it is also possible to measure the current during the constant movement with the scope. This measured motor current is exactly the value of FF Friction.

FF Acceleration

The parameter **FF Acceleration** helps the controller when accelerating by providing in anticipation a current proportional to the acceleration demanded. This parameter should be used wherever very fast and dynamic movements are needed, or where big load masses are connected. The value of this parameter can be calculated as follows:

FF Acceleration = (m / c_f)	FF Acc.	Feed forward [mA/(m/s^2)]
	m	Moved mass [g]
	c_f	Motor force constant [N/A]

where m is the moved mass (load mass + slider or stator mass) and c_f is the force constant of the chosen motor. The value for the force constant c_f can be read from the data sheets.

FF Deceleration

This parameter is the counterpart to **FF Acceleration** and is used for anticipatory control of the current while the motor is being braked. The value of this parameter can be calculated as follows:

FF Deceleration = (m / c_f)	FF Dec.	Feed forward [mA/(m/s^2)]
	m	Moved mass [g]
	c_f	Motor force constant [N/A]

where m is the moved mass[1] (load mass + slider or stator mass) and c_f is the force constant of the chosen motor. The value for the force constant c_f can be read from the data sheets.

7.6 Adjusting of the Current Offset

Current offset with horizontal moves

For applications with horizontal moves the circumstances for the forward and backward movement are identical and the parameter **Current Offset** should be zero.

Current offset with vertical moves

In applications with vertical moves the weight of the load mass leads to an asymmetrical controller behavior for the up and down moves. With the parameter **Current Offset** in the directory **\Drives\Drive X\Control Parameters** this asymmetry may be compensated. The value may be computed as follows:

Current offset= (m * g) /c_f	m	Load mass [kg]
	g	Gravitation 9.81 m/s^2
	c_f	Motor force constant [N/A]

The mass m is the moved mass (load mass + slider or stator mass). The force constant c_f can be read from the data sheets. The sign of the parameter **Current Offset** depends on the direction of the mounting. If the cable exit is in direction to the floor then the sign is positive otherwise its negative.

7.7 The Tuning Tool

The Tuning Tool was introduced in the software Release 1.3.10. It helps the user to calculate and set the **Feed Forward Parameters** and the **Current Offset** without need of reading the motor data sheet. The Tuning Tool is started by clicking the button “Show Tuning Tool” in the Parameter Inspector (see Figure below).

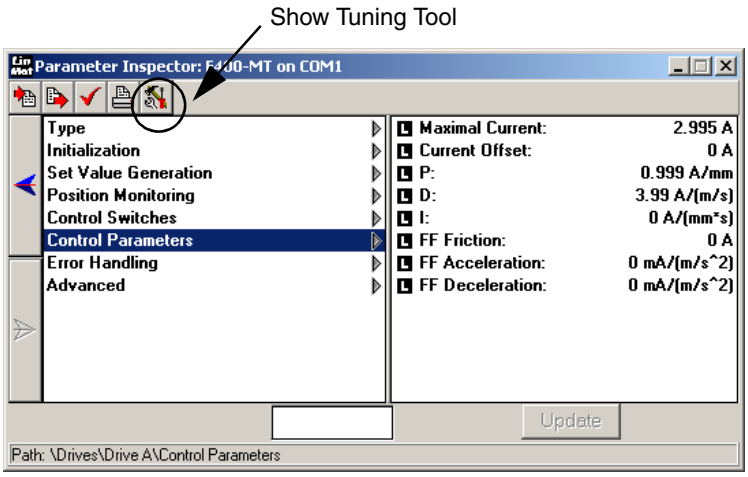


Figure 7-4: Parameter Inspector

Example: Using the Tuning Tool

A linear motor **P01-37x240/60x260** in vertically mounted (positive direction opposite to the gravity force) and moves a load mass of **1.2kg** attached to the slider. The linear motor has a force constant c_f of 40.8N/A and the slider mass is 829g. This sums up to a moved mass of 2029g.

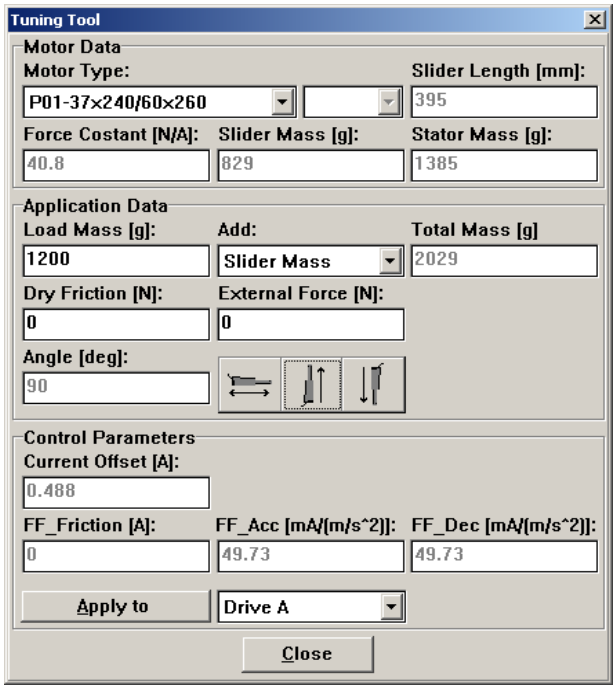


Figure 7-5: The Tuning Tool

Motor data

In the “Motor Type” pop-up menu select your motor type. If you have a special motor “F” (Fast type) or “S” (Short type) you can specify this in the pop-up menu on the right side of the motor type field. The slider length, slider mass, stator mass and force constant will be displayed.

Note:

If your motor is not in the motor type list then select the type “Other ...” and set the Force Constant, the Slider Mass and the Stator Mass in the appropriate fields (in this case you have to look up these values from the data sheet).

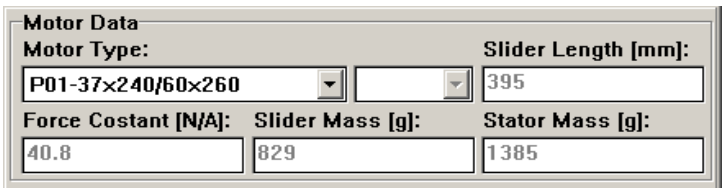


Figure 7-6: Choose the motor type

Application data

Set the load mass in the “Load Mass” field and select in the “Add” pop-up menu the moving part of the motor. The mass of the moving motor part will be added to the load mass. The total mass will be displayed in the “Total Mass” field. Set the dry friction in the “Dry Friction” field. If any external constant force exists (like MagSpring) set its value in the field “External Force”. The sign of this force is positive if it is in the same direction as the positive position direction of LinMot otherwise it is negative (see Figure 7-8, “Positive direction of LinMot motor”). Set the direction of the movement by clicking on the appropriate button. (see Figure 7-7, “Set application data”).

Note:

If there are more motors working in parallel in the master/booster or gantry mode, add the load mass and the mass of the moving part of all motors together and then divide this value by the number of motors. Set the resulting value in the “Load Mass” field and select “None” in the “Add” pop-up menu. With this method you can calculate and set the FF Parameter and the Current Offset for the master motor and all motors used for gantry!

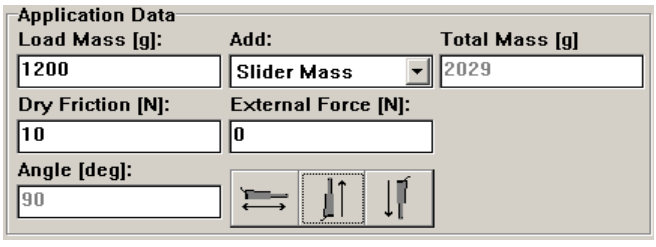


Figure 7-7: Set application data

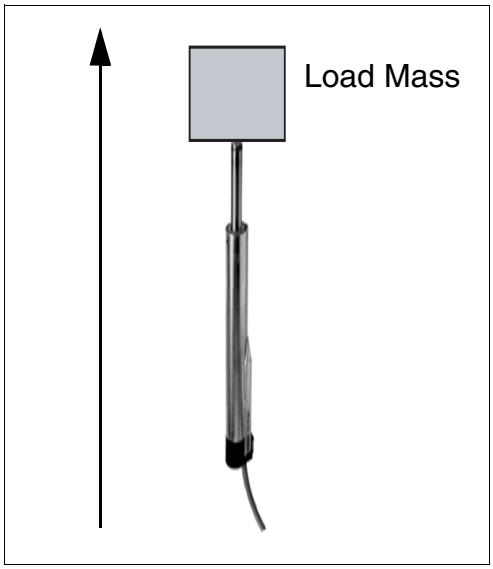


Figure 7-8: Positive direction of LinMot motor

Resulting control parameters values

The calculated values for the Feed Forward Parameters and the Current Offset will shown in the “Control Parameters” group (see Figure 7-9, “Resulting control parameters”). To take these values for the motor select the drive in the pop-up menu and press the button “Apply to”. If you are connected with a Controller these values will be written into it and they take immediately effect (live parameters).

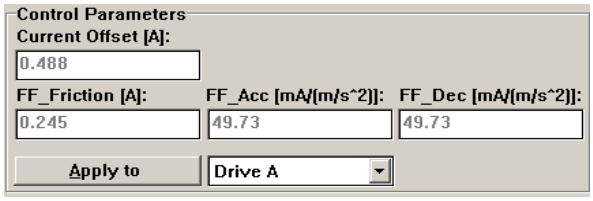


Figure 7-9: Resulting control parameters

Tip:

If the mass of an application is not known, the following method can be used: Stop the motor on a certain position. Reading out the needed motor current using the scope or reducing the max current down to the point where the motor can not longer hold the position.

7.8 Configuration of the max Current

The maximal current may be set with the parameter **Maximal Current** in the directory **\Drives\Drive X\Control Parameters**. The following values should be used

	Series E100		Series E1000 / E1001	
Motor type	**24V Supply**	**48V Supply**	**48V Supply**	**72 V Supply**
P01-23x80/…	2.0A	3.0A	3.0 A	3.0 A
P01-23x160/…	1.0A	2.0A	2.0A	2.8A
P01-37x120/…	—	3.0A	6.0A	6.0A
P01-37x240/…	—	3.0A	3.3A	5.0A

If smaller values are used the peak force according to the data sheets is reduced. Bigger values lead to unstable operation. Note that the current range of the E100 Servo Controller Series has to be switched (**\Drives\Drive X\Control Switches)** as well to adjust the max current.

7.9 Basic set up parameters for the Controller

The following settings can be used for general applications. Note that the current range of the E100 Servo Controller Series has to be switched (**\Drives\Drive X\Control Switches)** as well to adjust the max current.

	PS01 - 23x80	PS01- 23x160	PS01- 37x120	PS01- 37x240
Max Current	2.99 A	2 A	2.99 A	2.99 A
Current Offset	0 A	0 A	0 A	0 A
P	1 A/mm	1 A/mm	1 A/mm	1 A/mm
D	4 A/(m/s)	4 A/(m/s)	4 A/(m/s)	4 A/(m/s)
I	0 A/(mm*s)	0 A/(mm*s)	0 A/(mm*s)	0 A/(mm*s)
FF Friction	0 A	0 A	0 A	0 A
FF Acceleration	0 mA/(m/s^2)	0 mA/(m/s^2)	0 mA/(m/s^2)	0 mA/(m/s^2)
FF Deceleration	0 mA/(m/s^2)	0 mA/(m/s^2)	0 mA/(m/s^2)	0 mA/(m/s^2)
Filter max Speed	0.781 m/s	0.781 m/s	0.781 m/s	0.781 m/s
Filter max Accel.	30.5 m/s^2	30.5 m/s^2	30.5 m/s^2	30.5 m/s^2
Control Switches	3 A	3 A	3 A	3 A

Figure 7-10: Basic set up for E100 Controller with 48V supply

	PS01 - 23x80	PS01- 23x160	PS01- 37x120	PS01- 37x240
Max Current	4 A	2.8 A	6 A	5 A
Current Offset	0 A	0 A	0 A	0 A
P	1 A/mm	1 A/mm	1 A/mm	1 A/mm
D	4 A/(m/s)	4 A/(m/s)	4 A/(m/s)	4 A/(m/s)
I	0 A/(mm*s)	0 A/(mm*s)	0 A/(mm*s)	0 A/(mm*s)
FF Friction	0 A	0 A	0 A	0 A
FF Acceleration	0 mA/(m/s^2)	0 mA/(m/s^2)	0 mA/(m/s^2)	0 mA/(m/s^2)
FF Deceleration	0 mA/(m/s^2)	0 mA/(m/s^2)	0 mA/(m/s^2)	0 mA/(m/s^2)
Filter max Speed	0.781m/s	0.781m/s	0.781m/s	0.781m/s
Filter max Accel.	30.5 m/s^2	30.5 m/s^2	30.5 m/s^2	30.5 m/s^2

Figure 7-11: Basic set up for E1000 / E1001 Controller with 72V supply

7.10 Tuning of the controller

There are many very different ways of adjusting a PID controller. The following method has acquitted itself in practice:

1. Set phase current parameters according to chapter “Configuration of the max Current” on page 112
2. Set the filter parameters according to chapter “Adjusting of the prefilter (Filter)” on page 105 or create a profile according to chapter “Using profiles for reference position” on page 106
3. Set feed forward parameters according to chapter “Adjustment of the Feed-Forward Parameters” on page 107
4. After these two steps the following parameters in the list **\Drives\Drive X\Control Parameters** have to be set:

   P = 0.25 A/mm D = 2.00 A/(m/s) I = 0.00 A/(ms)

5. Now the desired motion profile can be loaded. Then the motor has to be started in the **Continuous Curve** mode.
6. Now increase the parameter **D** by 1.0 until the motor begins to oscillate. Then reduce the D value to 60%.
7. Now increase the parameter **P** by 0.25 until the motor begins to oscillate. Then reduce the P value to 80%.
8. The parameter **I** should only be set if the steady state position difference between the actual and demand position in standstill is to big.
   To set the parameter **I** increase the value by 5.0 until the steady state position difference is minimized and at the same time there is no overshoot when accelerating or decelerating.

7.11 Checking results

Correct adjustment of the controller is best verified with the oscilloscope integrated in the *LinMot*® talk software. Of prime importance is the comparison between the actual position and the target (demand) position.

In addition it is a good idea to check the required motor current with the scope. If the motor current stays in its limitation for to long it is a signal that the motor is overloaded and therefore proper tuning is not possible.

8. *LinMot*® ASCII protocol

8.1 Introduction

The *LinMot*® ASCII protocol offers the user the possibility of transmitting commands to the *LinMot*® servo controller with a simple ASCII protocol via an RS232 or RS485 interface. This enables a *LinMot*® servo controller to be integrated seamlessly in systems operated with the help of these standard interfaces.

The illustrations below show typical system environments in which *LinMot*® servo controllers are operated via an ASCII protocol.l

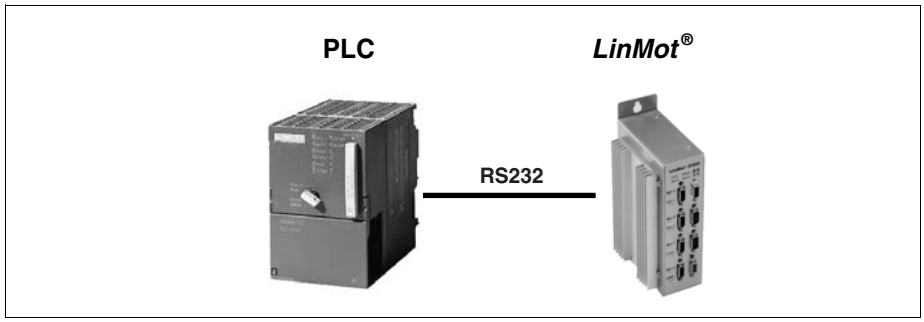


Figure 8-1: *LinMot*® servo controller with RS232

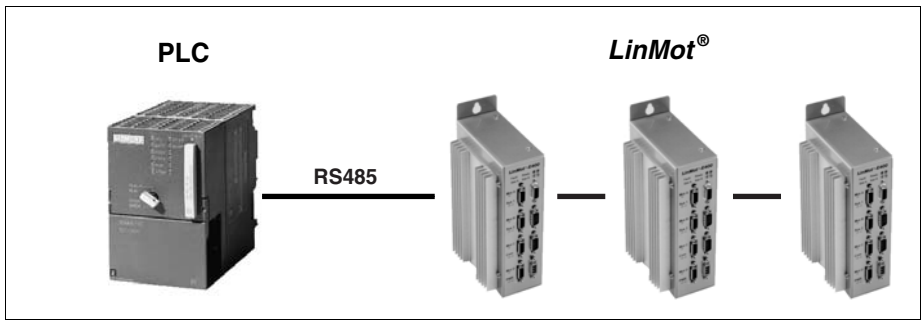


Figure 8-2: Several *LinMot*® servo controllers networked with RS485

8.2 Setup and installation

This section covers the setup and installation of a *LinMot*® servo controller for operation with the ASCII protocol.

Configuration

To control a *LinMot*® servo controller via the ASCII protocol, it must be configured appropriately. Configuration is performed with the help of the **Parameter Inspector** of the *LinMot*® talk software. When doing the setup, all parameters that **cannot be altered with the help of the ASCII protocol** have to be set:

- Command Interface
- Motor Type
- Run Mode
- Initialization Mode
- Error Handling

Provided these parameters are adjusted properly for the particular application and the servo controller is correctly linked with the PC or PLC, the *LinMot*® servo controller may be operated via the ASCII protocol.

Command Interface

This parameter is located in the **Parameter Inspector** under the path **\System\Command Interface**. It must be set either to **ASCII RS232** or **ASCII RS485**.

Motor Type

With these parameters each motor type can be specified. They are located under the path **\Drives\Drive X\Type**. The X stands for one of the maximum of 4 motors A to D.

Initialization Mode

The initialization mode of the motors is set by the parameters in the directory: **\Drives\Drive X\Initialization**.

RS232 operation

All *LinMot*® servo controllers are configured ex works for operation with the RS232 interface. The allocation of the interface is described in the user handbook. The interface is operated as follows:

Parameter	Value
Baud Rate	9’600
Start Bits	1
Data Bits	8
Stop Bits	1
Parity	No

If operating over RS232 the controller is always addressed with the bus ID = 1 independent from the position of the switches ID0 and ID1.

RS485 operation

For the operation over the RS485 link it is recommended to use the newer controller versions, where the code switches ID0 and ID1 are placed on the bottom side and not on the back side. The advantage is that the controllers have not to be opened for changing the jumpers.

If it is not avoidable to use a controller with the ID switches on the back side it is advisory to contact the *LinMot®* support (support@limot.com).

The RS485 link is configured as follows:

Parameter	Value
Baud Rate	9’600
Start Bits	1
Data Bits	8
Stop Bits	1
Parity	No

ID

The ID of the servo controller can be adjusted by means of the rotary switch on the bottom. When using the ASCII protocol via RS485 the ID numbers 1 to 6 are admissible for ID0 (ID LOW on version 3 controllers). ID1 (ID HIGH) must always be set to 0. Thus up to 6 servo controllers may be networked with RS485. The illustration below shows the servo controller from the bottom.

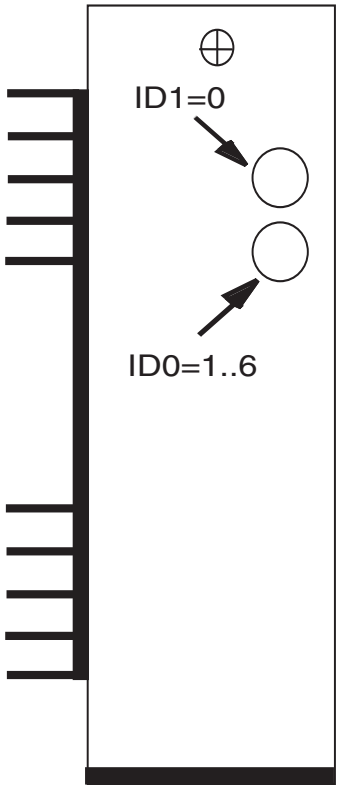


Figure 8-3: Adjusting the servo controller ID in RS485 mode

'Fail-save biasing'

With so-called fail-safe biasing, by using resistors an assured level is guaranteed on the bus even if no driver is active. This is necessary for proper functioning. The circuitry of the resistors may be seen from the diagram below.

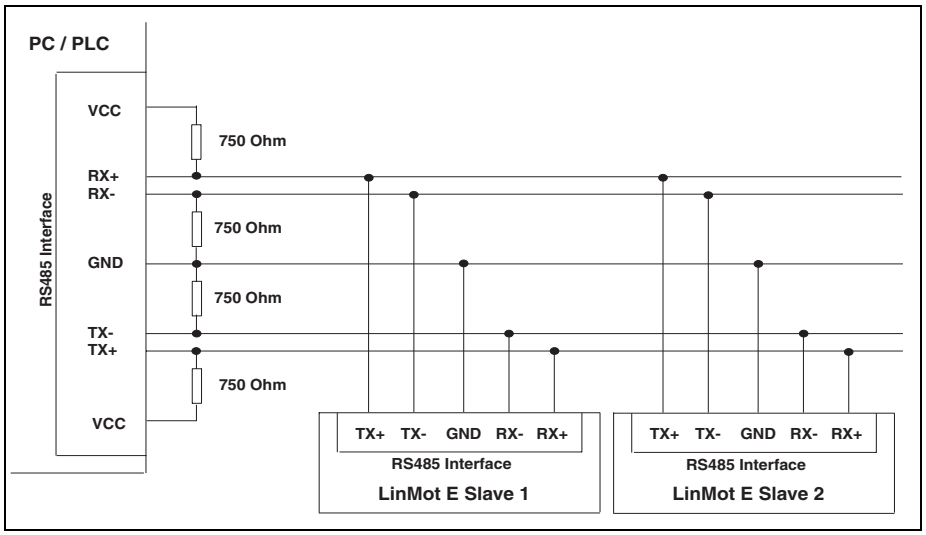


Figure 8-4: Half duplex cabling with RS485

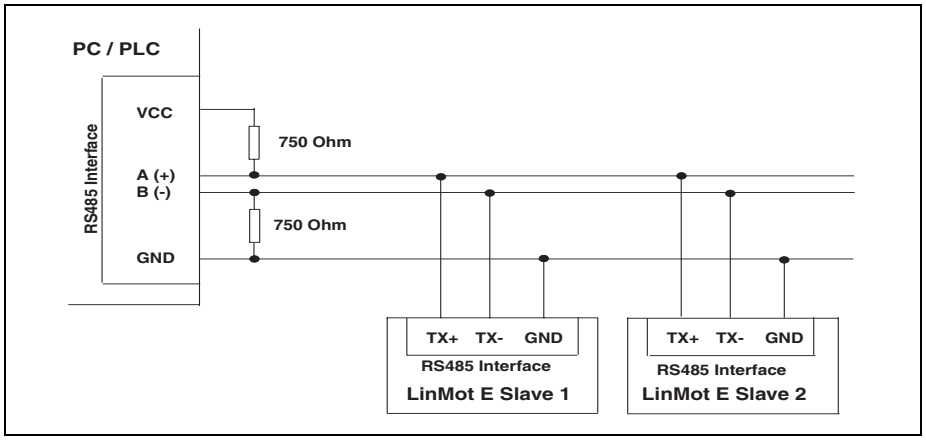


Figure 8-5: Half duplex cabling with RS485

Starting up

The following steps show how a servo controller is started up in conjunction with the ASCII protocol:

1. Configure the servo controller in accordance to the user manual chapter 5.1.2 (RS232) or chapter 5.1.3 (RS485). It is import to set the ID of the servo controller. The new ID is recognized only when the servo controller is powered up or started from the *LinMot®* Talk.
2. Start *LinMot®* Talk and parameterize servo controller in accordance to the user manual chapter 5.1.1.
3. Quit *LinMot®* Talk.
4. 4. Start using the ASCII protocol (e.g. with the Hyperterminal program of the windows platforms.

Warning You may not use the backspace or delete keys while using a terminal program to send commands to the servo controller!

8.3 Commands overview

ASCII Commands				L	S	M	E
	Value	**Set Command**	**Get Comand**	*LinMot*	Stepper	Magnet	System
Set Commands	Increment demand position	!IP	—	✘	✘		
	Increment demand position on next trigger	!TI	—	✘	✘		
	Set demand position on next trigger	!TP	—	✘	✘	✘	
	Goto position from actual postion	!SW	—	✘	✘	✘	
	Goto position from actual postion on next trigger	!TW	—	✘	✘	✘	
	Run curve	!RC	—	✘	✘	✘	
	Run curve on next trigger	!TC	—	✘	✘	✘	
	Run curve cyclic	!CC	—	✘	✘	✘	
	Run curve cyclic on next trigger	!CT	—	✘	✘	✘	
	Run curve profile at actual position	!RA	—	✘	✘	✘	
	Run curve on next trigger at actual position	!TA	—	✘	✘	✘	
	Run curve cyclic at actual position	!CA	—	✘	✘	✘	
	Run curve cyclic on trigger at actual position	!CB	—	✘	✘	✘	
	Stop cyclic motion profile	!CS	—	✘	✘	✘	
	Move home position	!MH	—	✘	✘		
	Redefine position	!RP	—	✘	✘		
	Set demand position to actual position	!RQ	—	✘	✘		
	Redefine position to zero	!ZD	—	✘	✘		
	Set Address Segment Offset	!AO	—				✘
	Set Address Segment Number	!AS	—				✘
	Write Memory Word	!WR	—				✘
	Write Memory Word with address increment	!WS	—				✘
Get/Set Commands	Demand position	!SP	!GD	✘	✘	✘	
	FF Acceleration	!DA	!EA	✘			
	FF Deceleration	!DB	!EB	✘			
	FF Friction	!DF	!EF	✘			
	P value of controller	!DP	!EP	✘			
	D value of controller	!DD	!ED	✘			
	I value of controller	!DI	!EI	✘			
	Maximal speed	!SV	!GV	✘	✘		
	Maximal acceleration	!SA	!GA	✘	✘		
	Maximal current	!SC	!GC	✘	✘	✘	
	Current offset	!DK	!EK	✘			
	Motion profile amplitude	!DC	!EC	✘	✘	✘	
	Motion profile offset	!DO	!EO	✘	✘	✘	
	Motion profile speed	!DS	!ES	✘	✘	✘	
	FREEZE flag	!SF	!GX	✘	✘	✘	✘
	INIT flag / **RUN** flag / **STOP** flag	!SI / !SR / !SS	!GX				✘

ASCII Commands				L	S	M	E
Get Commands	Actual current	—	!AC	✘			
	Actual position	—	!GP	✘	✘	✘	
	Position resolution	—	!PI	✘	✘	✘	
	Speed resolution	—	!VI	✘	✘		
	Acceleration resolution	—	!AI	✘	✘		
	Current resolution	—	!CI				✘
	State	—	!GS				✘
	Global error status	—	!GE				✘
	Global warn status	—	!GW				✘
	Motor error status	—	!EE	✘	✘	✘	
	Motor warn status	—	!EW	✘	✘	✘	
	State flags	—	!EX				✘
	Protocol version	—	!PV				✘
	Read Memory Word	—	!RD				✘
	Read Memory Word with address increment	—	!RE				✘

8.4 Command structure

All ASCII commands are structured to the following pattern:

Command structure		
Byte	**Value**	**Meaning**
0	'!'	Command head
1…2	char, char	Command
3…x	[char], …	Arguments
x+1	'↵' (0x0D)	End of command ASCII Character 0x0D (= 13 decimal) Carriage Return

Every command begins with an exclamation mark, followed by two characters coding the command, then the command arguments and finally a carriage return symbol.

Every command received on the *LinMot*® servo controller is acknowledged. A further command may be sent only if the last one has been acknowledged by the servo controller.

The command acknowledgement is structured as follows:

Acknowledge structure		
Byte	**Value**	**Meaning**
0	'#'	Acknowledge head
1…x	char, …	Acknowledge message
x+1	'↵' (0x0D)	End of acknowledge

The following example shows how a new demand position for motor A is given to the connected servo controller.

Example

Direction	ASCII sequence	Description
PC -> *LinMot*® controller	'!SP2000A' + 0x0D	Sets the demand position of motor A to 2000 increments.
LinMot® controller -> PC	'#' + 0x0D	When the '#'-symbol followed by a '↵'-symbol (0x0D) is transmitted, this means the command has been accepted by the *LinMot*® servo controller.

Hint

Over RS232 it is possible to run the ASCII and the LinMot-Talk protocol together. In this case it is important to finish all the commands and their answers completely before starting a command from the other protocol.

If the LinMot-Talk communication results in the "Device Time Out" error, it is most likely that the ASCII command was not finished correctly. In this case only a carriage returns (0x0D) has to be sent with a terminal program.

8.5 Commands

The table below provides an overview of the variable types employed in the following sections. The variable types may be subscripted and indicated in square brackets after the variable. Thus for example $pos_{[sint16]}$ stands for a variable named *pos* of type *sint16*. As may be seen from the table below, the variable type *sint16* stands for a signed cardinal value. Values in square brackets are optional. The '|' symbol stands for 'or'. In the

Argument types

Argument types

Argument type	Description
uint16	Cardinal value ranging 0 … 2^{16}-1. The value must be represented decimally ASCII-System. The plus sign is optional with positive values. Example: 0, 123, 3300, +200, 500
sint16	Signed cardinal value -2^{15} … 2^{15}-1. The value must be represented decimally in the ASCII system. The plus sign is optional with positive values. Example: 0, 123, -2000, -200, +240
uint32	Cardinal value ranging 0 … 2^{32}-1. The value must be represented decimally in the ASCII system. Example: 0, 123, 200000, 3000000, +240
ackcode	Acknowledge code sent back from the servo controller after every action or set command received. The individual codes are explained in chapter “Reference table: status and error messages” on page 150.
drivecode	Motor designator. Motor designators are ASCII capitals. If the protocol is set to RS232, only the motor designators ‘A’, ‘B’, ‘C’ and ‘D’ are allowed. Otherwise if the corresponding servo controller and motor exist, any designator may be used (‘A’…’X’). All designators are given in chapter “Reference table: motor designator” on page 152. Example: A … D => Drive A, B, C, D (controller with ID = 1) E … H => Drive A, B, C, D (controller with ID = 2) … U … X => Drive A, B, C, D (controller with ID = 6)
elocode	Servo controller designator. Servo controller designators are ASCII digits. If the protocol is set to RS232, only the digit ‘1’ is allowed. Otherwise digits ‘1’ to ‘6’ may be used provided the corresponding servo controller exist.
statecode	Status code sent back from the servo controller after a GS command. The individual codes are explained in chapter “Reference table: status and error messages” on page 150.

Command description

All commands in this chapter have been organized in alphabetic order. The following example shows how a description looks like.

ASCII COMMANDS All commands are shown and described in these tables. One table describes one ASCII command.

A —	SP Set demand position L S M	— C
B —	**Direction**	**ASCII sequence** — D
	PC → ELO	'!SP' + $pos_{[sint16]}$+ $drive_{[drivecode]}$ + 0x0D
	ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

A The heading row shows the command and a short command description

B Shows the direction of the communication.

C These icons specify on which motortypes the command works. Look also table 8-1, "Icon description".

D Shows the ASCII sequence.

The following table shows the meaning of the icons in the heading row of the command description.

Icon	Description
E	The command works for controllers
L	The command works for linear motors
S	The commands works for stepper motors
M	The command works for magnet / solenoids

Tabelle 8-1: Icon description

AC

AC Get demand current	L
Direction	**ASCII sequence**
PC → ELO	'!AC' + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $curr_{[sint16]}$ + 0x0D

This command queries the demand current set by the controller for the selected drive. The resolution of the current can be queried with the **CI** command.

Value	Min	Max
$curr_{[sint16]}$	-256	256

AI

AI Get acceleration resolution	E S
Direction	**ASCII sequence**
PC → ELO	'!AI' + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $accres_{[uint32]}$ + 0x0D

This command queries the acceleration resolution. The values are given back in various units depending on the motor type selected:

Motor type	Unit
LinMot®	1µm/s^2
Stepper motor	2^{-10} Steps/s^2

Typical sequence:

PC → ELO	ELO → PC	Description
!AIB↵	#238419↵	Queries actual acceleration resolution of motor B. The value given back corresponds to 0.238m/s^2, because motor B was configured as *LinMot®* in this case.

AO

AO Set address segment offset	E
Direction	**ASCII sequence**
PC → ELO	'!AO' + segoffset$_{[uint16]}$+ *drive*$_{[drivecode]}$ + 0x0D
ELO → PC	'#' + *ack*$_{[ackcode]}$ + 0x0D

This command sets the address segment offset for the read and write memory commands. The complete address is 24 bits wide and consists of the segment number (highest 8 bits) and the segment offset (lowest 16 bits). The address segment offset must be an even number. The drive indicator is used as a controller selector (any configured drive on the controller is possible).

Value	Min	Max
segoffset$_{[uint16]}$	0	65534

AS

AS Set address segment number	E
Direction	**ASCII sequence**
PC → ELO	'!AS' + seg$_{[uint8]}$+ *drive*$_{[drivecode]}$ + 0x0D
ELO → PC	'#' + *ack*$_{[ackcode]}$ + 0x0D

This command sets the address segment number for the read and write memory commands. The complete address is 24 bits wide and consists of the segment number (highest 8 bits) and the segment offset (lowest 16 bits). The drive indicator is used as a controller selector (any configured drive on the controller is possible).

Value	Min	Max
seg$_{[uint8]}$	0	255

CA

CA Run curve cyclic at actual position	L S M
Direction	**ASCII sequence**
PC → ELO	'!CA' + *curve*$_{[uint16]}$ + *drive*$_{[drivecode]}$ + 0x0D
ELO → PC	'#' + *ack*$_{[ackcode]}$ + 0x0D

This command runs a curve cyclic from the actual wanted position. This means that as soon as the motion profile is done it will be run again. With the help of the **CS** command the motion profile can be stopped. As long as the motion profile is running no other motion commands may be executed on the motor. This command must only be executed in the **RUN** state.

Note: This command will change the Curve Position Offset parameter.

CB

CB Run curve cyclic on next trigger at actual position L S M	
Direction	**ASCII sequence**
PC → ELO	'!CB' + $curve_{[uint16]}$ + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command starts with the next positive edge of the trigger signal a motion profile cyclical from the actual wanted position. After the first start the motion profile will be run cyclic without need for a fresh trigger signal. With the help of the **CS** command the motion profile can be stopped. As long as the motion profile is running no other motion commands may be executed on the motor. This command must only be executed in the **RUN** state.

Note: This command will change the Curve Position Offset parameter.

CC

CC Run curve cyclic L S M	
Direction	**ASCII sequence**
PC → ELO	'!CC' + $curve_{[uint16]}$ + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command runs a curve cyclic. This means that as soon as the motion profile is done it will be run again. With the help of the **CS** command the motion profile can be stopped. As long as the motion profile is running no other motion commands may be executed on the motor. This command must only be executed in the **RUN** state.

CI

CI Get current resolution E	
Direction	**ASCII sequence**
PC → ELO	'!CI' + $elo_{[elocode]}$ + 0x0D
ELO → PC	'#' + $currres_{[uint32]}$ + 0x0D

This command queries the current resolution of the selected servo controller. The values are given back in µA.

Typical sequence:

PC → ELO	ELO → PC	Description
!CI1↵	#23438↵	Queries current resolution of servo controller with ID 1. The value given back corresponds to 23.438mA.

CS

CS Stop cyclic motion profile L S M

Direction	ASCII sequence
PC → ELO	'!CS' + $stop_{[drivesel]}$ + $anydrive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command terminates the execution of one or more cyclic motion profiles. The motion profile run to their end after reception of this command and will not be started again. The variable *drivesel* specifies which motors should be halted:

Bit	3	2	1	0
Name	Motor D	Motor C	Motor B	Motor A

With the variable *anydrive* a random motor may be chosen on the servo controller.

Example sequence:

PC → ELO	ELO → PC	Description
!CS3A↵	#↵	Stops the execution of the cyclic motion profiles for the motors A and B.

CT

CT Run motion profile cyclic on next trigger L S M

Direction	ASCII sequence
PC → ELO	'!CT' + $curve_{[uint16]}$ + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command starts with the next positive edge of the trigger signal a motion profile cyclical. After the first start the motion profile will be run cyclic without need for a fresh trigger signal. With the help of the **CS** command the motion profile can be stopped. As long as the motion profile is running no other motion commands may be executed on the motor. This command must only be executed in the **RUN** state.

DA

DA Set FF Acceleration value of controller L

Direction	ASCII sequence
PC → ELO	'!DA' + $ffacc_{[uint16]}$ + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command sets the **FF Acceleration** value of the motor *drive* to value *ffacc*. The unit is 0.1 mA/(m/s^2).

Value	Min	Max
$ffacc_{[uint16]}$	0	32640

DB

DB Set FF Deceleration value of controller	L
Direction	**ASCII sequence**
PC → ELO	'!DB' + $ffdec_{[uint16]}$ + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command sets **FF Deceleration** value of the motor *drive* to *ffdec*. The unit is 0.1 mA/(m/s^2).

Value	Min	Max
$ffdec_{[uint16]}$	0	32640

DC

DC Set motion profile amplitude	L S M
Direction	**ASCII sequence**
PC → ELO	'!DC' + $camp_{[uint16]}$ + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command set the motion profile amplitude of the chosen motor. The maximum value (4096) is equal to the scale factor 100%. With this value the amplitude of the motion profile is as big as it was defined in the **Curve Inspector**.

Warning: The motor will jump if this command is used while a curve is running.

Value	Min	Max
$camp_{[uint16]}$	0	4096

DD

DD Set D value of controller	L
Direction	**ASCII sequence**
PC → ELO	'!DD' + $d_{[uint16]}$ + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command sets the **D** value of the motor *drive* to the value *dec*. The unit is 0.015 A/(m/s).

Value	Min	Max
$d_{[uint16]}$	0	32640

DF

DF Set FF Friction value of controller	L
Direction	**ASCII sequence**
PC → ELO	'!DF' + $fffri_{[uint16]}$ + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command sets the **FF Friction** value of the motor *drive* to the value *fffri*. The unit is 0.0234 A.

Value	Min	Max
$fffri_{[uint16]}$	0	255

DI

DI Set I value of controller	L
Direction	**ASCII sequence**
PC → ELO	'!DI' + $i_{[uint16]}$ + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command sets the **I** value of the motor *drive* to the value *i*. The unit is 0.0457 A/(mm*s).

Value	Min	Max
$i_{[uint16]}$	0	32640

DK

DK Set current offset	L
Direction	**ASCII sequence**
PC → ELO	'!DK' + $curroff_{[uint16]}$ + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This commands sets the current offset of the motor *drive* to the value *curroff*. The current resolution may be queried with the **CI** command. Adjusting the current offset is reasonable if the motor is mounted vertically and is loaded with changing load masses.

Value	Min	Max
$curroff_{[uint16]}$	-256	256

DO

DO Set motion profile offset	L S M
Direction	**ASCII sequence**
PC → ELO	'!DO' + $coff_{[sint16]}$ + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command sets the motion profile offset of the selected drive. The position resolution may be queried with the **PI** command.

Value	Min	Max
$coff_{[sint16]}$	-32256	+32256

DP

DP Set P value of controller	L
Direction	**ASCII sequence**
PC → ELO	'!DP' + $p_{[uint16]}$ + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command sets the **P** value of the motor *drive* to *p*. The unit is 0.00234 A/mm.

Value	Min	Max
$p_{[uint16]}$	0	32640

DS

DS Set motion profile speed	L S M
Direction	**ASCII sequence**
PC → ELO	'!DS' + $vel_{[uint16]}$ + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command sets the speed of motion profiles. When the maximum value is set the motion profile is run as fast as it has been designed. With lower values the speed drops linearly. The speed may be altered any time.

Value	Min	Max
$vel_{[uint16]}$	0	4096

EA

EA Get FF Acceleration value of controller	L
Direction	**ASCII sequence**
PC → ELO	'!EA'+ $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ffacc_{[uint16]}$ + 0x0D

This command gives the **FF Acceleration** value of the motor *drive*. The unit is 0.1 mA/(m/s^2).

Value	Min	Max
$ffacc_{[uint16]}$	0	32640

EB

EB Get FF Deceleration value of controller	L
Direction	**ASCII sequence**
PC → ELO	'!EB' + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ffdec_{[uint16]}$ + 0x0D

This command gives the **FF Deceleration** value of the motor *drive*. The unit is 0.1 mA/(m/s^2).

Value	Min	Max
$ffdec_{[uint16]}$	0	32640

EC

EC Get motion profile amplitude	L S M
Direction	**ASCII sequence**
PC → ELO	'!EC' + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $camp_{[uint16]}$ + 0x0D

This command gets the motion profile amplitude of the selected motor. The maximum value (4096) is equal to 100%.

Value	Min	Max
$camp_{[uint16]}$	0	4096

ED

ED Get D value of controller	L
Direction	**ASCII sequence**
PC → ELO	'!ED' + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $d_{[uint16]}$ + 0x0D

This command gives back the **D** value of the motor *drive*. The unit is 0.015A/(m/s)

Value	Min	Max
$d_{[uint16]}$	0	32640

EE

EE Get motor error status	L S M
Direction	**ASCII sequence**
PC → ELO	'!EE' + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $state_{[statecode]}$ + 0x0D

This command gets the actual **motor error status** of the selected motor. The bits have the following meaning:

Bit	7	6	5	4	3	2	1	0
Name	no / incompatible motion profile	wrong motortype	init failed	reserved	slider missing	following error	motor hot sensor	motor hot calculated

EF

EF Get FF Friction value of controller	L
Direction	**ASCII sequence**
PC → ELO	'!EF' + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $fffri_{[uint16]}$ + 0x0D

This command gives back the **FF Friction** value of the motor *drive*. The current resolution may be queried with the **CI** command.

Value	Min	Max
$fffri_{[uint16]}$	0	255

EI

EI Get I value of controller	L
Direction	**ASCII sequence**
PC → ELO	'!EI' + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $i_{[uint16]}$ + 0x0D

This command gives back the **I** value of the motor *drive*. The unit is 0.0457A/(mm*s).

Value	Min	Max
$i_{[uint16]}$	0	32640

EK

EK Get current offset	L
Direction	**ASCII sequence**
PC → ELO	'!EK' + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $curroff_{[uint16]}$ + 0x0D

This command queries the current offset of the motor *drive*. The resolution of the current may be queried with the **CI** command.

Value	Min	Max
$curroff_{[uint16]}$	-256	256

EO

EO Get motion profile offset	L S M
Direction	**ASCII sequence**
PC → ELO	'!EO' + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $coff_{[sint16]}$ + 0x0D

This command gets the motion profile offset of the selected motor. The position resolution may be queried with the **PI** command.

Value	Min	Max
$coff_{[sint16]}$	-32256	+32256

EP

EP Get P value of controller	L
Direction	**ASCII sequence**
PC → ELO	'!EP' + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $p_{[uint16]}$ + 0x0D

This command gives back the **P** value of the motor *drive*. The unit is 0.00234 A/mm.

Value	Min	Max
$p_{[uint16]}$	0	32640

ES

ES Get motion profile speed	L S M
Direction	**ASCII sequence**
PC → ELO	'!ES' + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $vel_{[uint16]}$ + 0x0D

This command gives back the speed of the motion profiles. If the maximum value is set the motion profile is run as fast as it has been designed. With lower values the speed drops linearly.

Value	Min	Max
$vel_{[uint16]}$	0	4096

EW

EW Get motor warn status	L S M
Direction	**ASCII sequence**
PC → ELO	'!EW' + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $state_{[statecode]}$ + 0x0D

This command gets the actual **motor warn status** of the selected motor. The bits have the following meaning:

Bit	8	7	6	5	4	3	2	1	0
Name	motor not in defined position range	reserved	initialization not yet done	reserved	reserved	slider missing	following error	motor hot sensor	motor hot calculated

EX

EX Get state flags	E
Direction	**ASCII sequence**
PC → ELO	'!EX' + $elo_{[elocode]}$ + 0x0D
ELO → PC	'#' + $state_{[statecode]}$ + 0x0D

This command gets the state flags of the selected servo controller. The bits have the following meaning:

Bit	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	**EMERG STOP** substate flag	**INIT** state	**DISABLE** state	**ERROR** state	**RUN** state	**INIT NOT DONE** flag	**WARN** flag	**ERROR** flag	In Position Drive D	In Position Drive C	In Position Drive B	In Position Drive A

Typical sequence:

PC → ELO	**ELO → PC**	**Description**
!EX1↵	#129↵	The read value shows that the servo controller is in the state **RUN** and the motor A is in position.

GA

GA Get maximum acceleration	L S
Direction	**ASCII sequence**
PC → ELO	'!GA' + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $acc_{[uint16]}$ + 0x0D

This command gives back the maximum acceleration of the *drive* motor. The resolution of the acceleration may be queried with the **AI** command.

Value	**Min**	**Max**
$acc_{[uint16]}$	1	1536

GC

GC Get maximum current (force)	L S M
Direction	**ASCII sequence**
PC → ELO	'!GC' + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $curr_{[uint16]}$ + 0x0D

This command gives the maximum current of the motor *drive*. The resolution may be queried with the **CI** command.

Value	Min	Max
$curr_{[uint16]}$	0	256

GD

GD Get demand position	L S M
Direction	**ASCII sequence**
PC → ELO	'!GD' + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $pos_{[sint16]}$ + 0x0D

This command gives back the target (demand) position of the motor *drive*. This command may not be used when a motion profile is run. The resolution of the position may be queried with the command **PI**.

Value	Min	Max
$pos_{[sint16]}$	-32256	32256

GE

GE Get global error status	E
Direction	**ASCII sequence**
PC → ELO	'!GE' + $elo_{[elocode]}$ + 0x0D
ELO → PC	'#' + $state_{[statecode]}$ + 0x0D

This command gets the actual **global error status** of the selected motor. The bits have the following meaning:

Bit	7	6	5	4	3	2	1	0
Name	reserved	reserved	controller fault[1]	DCLV signal[2] too high	DCLV signal[2] too low	DCLV power[3] too high	DCLV power[3] too low	reserved

1) The heat sink of the servo controller is too hot (over 70° celsius) or a short circuit on a motor phase has been detected.
2) **DCLV signal** stands for the DC link voltage of the signal board.
3) **DCLV Power** stands for the DC link voltage of the power board.

GP

GP Get actual position	L S M
Direction	**ASCII sequence**
PC → ELO	'!GP' + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $pos_{[sint16]}$ + 0x0D

This command queries the actual position the motor drive. The resolution of the position may be queried with the command **PI**.

Typical sequence:

PC → ELO	ELO → PC	Description
!GPA↵	#256↵	Queries the actual position of motor A.

GS

GS Get actual state	E
Direction	**ASCII sequence**
PC → ELO	'!GS' + $elo_{[elocode]}$ + 0x0D
ELO → PC	'#' + $state_{[statecode]}$ + 0x0D

This command gives the actual state of the servo controller. The state consists of one letter which encodes the state, and a number. The number is transmitted only in the error state and encodes the actual error. The status coding is explained in chapter "Reference table: status and error messages" on page 150.

Typical sequence:

PC → ELO	ELO → PC	Description
!GS1↵	#R↵	Queries the actual state. The 'R' denotes that the servo controller is in the **RUN** state.

GV

GV Get maximum speed	L S
Direction	**ASCII sequence**
PC → ELO	'!GV' + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $vel_{[uint16]}$ + 0x0D

This command gives back the maximum speed of the motor *drive*. The resolution of the speed may be queried with the command **VI**.

Value	Min	Max
$vel_{[uint16]}$	6	24576

GW

GW Get global warn status E

Direction	ASCII-Sequence
PC → ELO	'!GW' + $elo_{[elocode]}$ + 0x0D
ELO → PC	'#' + $state_{[statecode]}$ + 0x0D

This command gets the **global warn status** of the selected servo controller. The bits have the following meaning:

Bit	7	6	5	4	3	2	1	0
Name	reserved	reserved	controller warning[1]	DCLV signal[2] high	DCLV signal low	DCLV power[3] high	DCLV power low	reserved

1) The heat sink of the servo controller is too hot (over 70° celsius) or a short circuit on a motor phase has been detected. After 5 seconds the controller goes into the **ERROR** state.
2) **DCLV Signal** stands for the DC link voltage of the signal board.
3) **DCLV Power** stands for the DC link voltage of the power board.

GX

GX Get flags E

Direction	ASCII sequence
PC → ELO	'!GX' + $elo_{[elocode]}$ + 0x0D
ELO → PC	'#' + $flags_{[flagcode]}$ + 0x0D

This command gets all flags of the selected servo controller. The bits have the following meaning:

Bit	7	6	5	4	3	2	1	0
Name	**RUN** Flag	**INIT** Flag	**FREEZE** Flag (global)	**STOP** Flag	**FREEZE** Flag Drive D	**FREEZE** Flag Drive C	**FREEZE** Flag Drive B	**FREEZE** Flag Drive A

Typical sequence:

PC → ELO	ELO → PC	Description
!GX1↵	#134↵	The read value shows that the **RUN** flag and the **FREEZE** flags for the motors B and C are set.

IP

IP Increment demand position (relative positioning) L S	
Direction	**ASCII sequence**
PC → ELO	'!IP' + $posinc_{[sint16]}$ + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command increments the target (demand) position of the motor *drive* by the amount *posinc*. This command must only be executed in the **RUN** state. The resolution of the position may be queried with the command **PI**. The increment position must be in the range of -32256 to 32256.

Typical sequence:

PC → ELO	ELO → PC	Description
!IP100A↵	#↵	Increments the target position by 100 units.

MH

MH Move home position L S	
Direction	**ASCII sequence**
PC → ELO	'!MH' + $posinc_{[sint16]}$ + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command moves the home position of the motor by the amount *posinc*. It has purpose only in few cases and should be used with caution. This command must only be executed in the **RUN** state. Important: The minimum and maximum positions are **not** displaced as well!

PI

PI Get position resolution L S M	
Direction	**ASCII sequence**
PC → ELO	'!PI' + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $posinc_{[uint32]}$ + 0x0D

This command queries the actual position resolution. The values are given in different units depending on the motor type selected:

Motor type	Unit
LinMot®	1 pm (10^{-12} m)
Stepper motor	1/256 Step
Solenoid	1 µA

Typical sequence:

PC → ELO	ELO → PC	Description
!PIA↵	#19531250 ↵	The return value '19531250' means the position resolution is 19.531250µm (if 'LinMot' is selected as motor type for motor A). The command !IP1000A' will therefore move motor 'A' by 1000 * 19.53µm.

PV

PV Get protocol version	E
Direction	**ASCII sequence**
PC → ELO	'!PV' + $elo_{[elocode]}$ + 0x0D
ELO → PC	'#' + $version_{[uint16]}$ + 0x0D

This command queries the actual version number of the protocol. All implemented commands in Release 1.3 correspond to protocol version 2. The commands in the protocol version 2 are a super set of the commands in protocol version 1 (Release 1.2).

Typical sequence:

PC → ELO	ELO → PC	Description
!PV1↵	#2↵	Queries the actual protocol version.

RA

RA Run curve at actual position	L S M
Direction	**ASCII sequence**
PC → ELO	'!RA' + $curve_{[uint16]}$+ $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command starts the stored motion profile from the actual wanted position. The *curve* parameter defines the profile which is to be started. The admissible range goes from 0 to 63, with 0 standing for an empty motion profile. This command must only be executed in the **RUN** state.

Note: When using this command the Curve Position Offset parameter will be changed. If the RA command is given while another motion profile is already being run, the new profile is started at once.

RC

RC Run motion profile	L S M
Direction	**ASCII sequence**
PC → ELO	'!RC' + $curve_{[uint16]}$+ $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command starts the stored motion profile. The *curve* parameter defines the profile which is to be started. The admissible range goes from 0 to 63, with 0 standing for an empty motion profile. This command must only be executed in the **RUN** state.

If the RC command is given while another motion profile is already being run, the new profile is started at once.

RD

RD Read memory word	E
Direction	**ASCII sequence**
PC → ELO	'!RD' + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $data_{[uint16]}$ + 0x0D

This command reads a memory word (16 bits) from the address previously set with the AO and AS commands. The drive indicator is used as a controller selector (any configured drive on the controller is possible).

Note: this address can be altered by using the commands WS or RE.

Value	Min	Max
$data_{[uint16]}$	0	65535

RE

RE Read memory word with address increment	E
Direction	**ASCII sequence**
PC → ELO	'!RE' + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $data_{[uint16]}$ + 0x0D

This command reads a memory word (16 bits) from the address previously set with the AO and AS commands. After reading the address will be incremented automatically by 2. The drive indicator is used as a controller selector (any configured drive on the controller is possible).

Value	Min	Max
$data_{[uint16]}$	0	65535

RP

RP Redefine actual position	L S
Direction	**ASCII sequence**
PC → ELO	'!RP' + $pos_{[sint16]}$ + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command sets the actual position of the motor *drive* to the value *pos*. It has purpose in only few cases and should be used with caution. Important: The minimum and maximum positions are **not** redefined!

This command must only be executed in the **RUN** state.

Value	Min	Max
$pos_{[sint16]}$	-32256	32256

RQ

RQ Set demand position to actual position	L S
Direction	**ASCII sequence**
PC → ELO	'!RQ' + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command sets the demand position to the acutal position. This command is used e.g. when the motor has been freezed and the motion should not continue when releasing from freeze if the motor has been current free and should be powered again without moving.

SA

SA Set maximum acceleration	L S
Direction	**ASCII sequence**
PC → ELO	'!SA' + $acc_{[uint16]}$ + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command sets the maximum acceleration of the drive motor to the value *acc*. The resolution of the acceleration may be queried with the command **AI**.

Value	Min	Max
$acc_{[uint16]}$	1	1536

SC

SC Set maximum current (force)	L S M
Direction	**ASCII sequence**
PC → ELO	'!SC' + $curr_{[uint16]}$ + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command sets the maximum current of the *drive* motor to the value *curr*. The current resolution may be queried with the command **CI**.

Value	Min	Max
$curr_{[uint16]}$	0	256

SF

SF Set FREEZE flags	L S M E
Direction	**ASCII sequence**
PC → ELO	'!SF' + ('+'|'-') + ($elo_{[elocode]}$|$drive_{[drivecode]}$) + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This commands sets ('+') or clears ('-') the **FREEZE** flags. On each servo controller there is a global **FREEZE** flag, which freezes the motion of all drives and four motor specific **FREEZE** flags, which freeze the motion of a single motor. A motor moves only if both the global **FREEZE** flag and his motor specific **FREEZE** flag are cleared.

Typical sequence:

PC → ELO	ELO → PC	Description
!SF+A↵	#↵	Sets the **FREEZE** flag on motor A
!SF+1↵	#↵	Sets the global **FREEZE** flag of the servo controller with the ID 1

SI

SI Set INIT flag	E
Direction	**ASCII sequence**
PC → ELO	'!SI' + ('+'|'-') + $elo_{[elocode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command sets ('+') or clears ('-') the **INIT** flag. The meaning of the **INIT** flag is given in chapter "Operational states" on page 11.

SP

SP Set demand position (absolute positioning)	L S M
Direction	**ASCII sequence**
PC → ELO	'!SP' + $pos_{[sint16]}$+ $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command sets the demand position for the motor *drive* to the value *pos*. The resolution of the position may be queried with the command **PI**.

Value	Min	Max
$pos_{[sint16]}$	-32256	32256

SR

SR Set RUN flag	E
Direction	**ASCII sequence**
PC → ELO	'!SR' + ('+'|'-') + $elo_{[elocode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command sets ('+') or clears ('-') the **RUN** flag. The meaning of the **RUN** flag is described in chapter "Operational states" on page 11.

SS

SR Set STOP flag	E
Direction	**ASCII sequence**
PC → ELO	'!SS' + ('+'|'-') + $elo_{[elocode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command sets ('+') or clears ('-') the **STOP** flag. The meaning of the **STOP** flag is described in chapter "Operational states" on page 11.

SV

SV Set maximum speed	L S
Direction	**ASCII sequence**
PC → ELO	'!SV' + $vel_{[uint16]}$ + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command sets the maximum speed of the motor *drive* to the value *vel*. The resolution of the speed may be queried with the command **VI**.

Value	Min	Max
$vel_{[uint16]}$	6	24576

SW

SW Goto position from actual position (absolute positioning)	L S M
Direction	**ASCII sequence**
PC → ELO	'!SW' + $pos_{[sint16]}$+ $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command sets the demand position for the motor *drive* to the value *pos*. The resolution of the position may be queried with the command **PI**. In contrast to the !SP command the V/A-limiter starts at the actual position. This command can be used for releasing from a press situation.

Value	Min	Max
$pos_{[sint16]}$	-32256	32256

TA

TA Run curve on next trigger at actual position	L S M
Direction	**ASCII sequence**
PC → ELO	'!TA' + $curve_{[uint16]}$+ $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command runs a stored motion profile from the actual wanted position on the next trigger pulse. Since the servo controller has very fast trigger inputs, a motion profile can be started very precisely. The motion profiles are always started on the positive slope. The *curve* parameter defines the motion profile that is to be started. The admissible range goes from 0 to 63, with 0 always standing for an empty motion profile. This command must only be executed in the **RUN** state.

Note: When using this command the Curve Position Offset parameter will be changed.

TC

TC Run motion profile on next trigger	L S M
Direction	**ASCII sequence**
PC → ELO	'!TC' + $curve_{[uint16]}$+ $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command runs a stored motion profile on the next trigger pulse. Since the servo controller has very fast trigger inputs, a motion profile can be started very precisely. The motion profiles are always started on the positive slope. The *curve* parameter defines the motion profile that is to be started. The admissible range goes from 0 to 63, with 0 always standing for an empty motion profile. This command must only be executed in the **RUN** state.

TI

TI Increment demand position on next trigger	L S
Direction	**ASCII sequence**
PC → ELO	'!TI' + $posinc_{[sint16]}$+ $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command increments the demand position of the motor *drive* by the amount *posinc* on the next trigger. The move is always started on the rising edge of the signal. This command must only be executed in the **RUN** state. The increment position must be in the range from -32256 to 32256.

TP

TP Set demand position on next trigger	L S M
Direction	**ASCII sequence**
PC → ELO	'!TP' + $pos_{[sint16]}$+ $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command sets the demand position of the motor *drive* to the *pos* value at the next trigger pulse. Since the servo controller has very fast trigger inputs, a movement can be started very precisely. The movement is always started on the positive edge. This command must only be executed in the **RUN** state.

Value	Min	Max
$pos_{[sint16]}$	-32256	32256

TW

TW Goto position from actual position on next trigger (absolute positioning)	L S M
Direction	**ASCII sequence**
PC → ELO	'!TW' + $pos_{[sint16]}$+ $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command sets the demand position of the motor *drive* to the *pos* value at the next trigger pulse. Since the servo controller has very fast trigger inputs, a movement can be started very precisely. The movement is always started on the positive edge. This command must only be executed in the **RUN** state. In contrast to the !TP command the V/A-limiter starts at the actual position. This command can be used for releasing froma press situation.

Value	Min	Max
$pos_{[sint16]}$	-32256	32256

VI

VI Get actual speed resolution	L S
Direction	**ASCII sequence**
PC → ELO	'!VI' + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $velres_{[uint32]}$ + 0x0D

This command queries the actual speed resolution. The values are given in different units depending on the motor type selected.

Motor type	Unit
LinMot®	1 nm/s
Stepper motor	2^{-16} Steps/s
Solenoid	0

Typical sequence:

PC → ELO	ELO → PC	Description
!VIA↵	#190735↵	Queries actual speed resolution motor A. The value given corresponds to 0.190735 mm/s, because in this example motor A is configured as *LinMot®*.

WR

WR Write memory word	E
Direction	**ASCII sequence**
PC → ELO	'!WR' + $data_{[uint16]}$ + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command writes a memory word (16 bits) at the address previously set with the AO and AS commands. The drive indicator is used as a controller selector (any configured drive on the controller is possible)

Note: this address can be altered by using the commands WS or RE.

Value	Min	Max
$data_{[uint16]}$	0	65535

WS

WS Write memory word with address increment	E
Direction	**ASCII sequence**
PC → ELO	'!WS' + $data_{[uint16]}$ + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command writes a memory word (16 bits) at the address previously set with the AO and AS commands. After writing, the address will be incremented automatically by 2. The drive indicator is used as a controller selector (any configured drive on the controller is possible).

Value	Min	Max
$data_{[uint16]}$	0	65535

ZD

ZD Set internal position counter to zero	L S
Direction	**ASCII sequence**
PC → ELO	'!ZD' + $drive_{[drivecode]}$ + 0x0D
ELO → PC	'#' + $ack_{[ackcode]}$ + 0x0D

This command sets the internal position counter to 0, shifting the home position at the same time. This command has purpose only in few cases and should be used with caution. Important: The minimum and maximum positions are not displaced as well! This command must only be executed in the **RUN** state.

8.6 Typical sequence

The following typical sequence provides an overview of the capabilities of the ASCII protocol. The symbol ↵ denotes the hexadecimal value 0x13. To repeat this sequence the motor connectors "Mot A" and "Mot B" must each be linked to a *LinMot®* motor and configured. Moreover at least one motion profile must be stored in the servo controller.

Typical sequence:

PC → ELO	ELO → PC	Description
!PV1↵	#2↵	Queries the actual protocol version of servo controller.
!GS1↵	#W↵	Queries the state of servo controller ‘1’. The ‘W’ denotes the Wait for Disable state.
!SR-1↵	#↵	Clear RUN request flag.
!SI-1↵	#↵	Clear INIT request flag.
!GS1↵	#D↵	Queries the state of servo controller ‘1’. The ‘D’ denotes the Disable state.
!SI+1↵	#↵	Set INIT request flag. Initialization starts.
!EX1↵	#1024↵	Wait until Init Not Done flag is cleared. The denoted state is still INIT. This command may be repeated until the initialization has finished.
!SR+1↵	#W↵	Set RUN request flag.
!SI-1↵	#↵	Clear INIT request flag.
!GS1↵	#R↵	Queries the state of servo controller ‘1’. The ‘R’ denotes the RUN state.
!CI1↵	#23438↵	Queries the current increment of servo controller ‘1’. The value signaled back corresponds to 23.439 mA.
!SC43B↵	#↵	Sets maximum current for motor ‘B’ to 1 A.
!SC64A↵	#↵	Sets maximum current for motor ‘A’ to 1.5 A.
!ZDB↵	#↵	Resets value of internal position counter of motor ‘B’ to 0.
!PIA↵	#19531250↵	Queries position increment of motor ‘A’. The value signaled back corresponds to 19.53125 µm.
!SP2560B↵	#↵	Sets the demand position of drive ‘B’ to 50 mm.
!RC1A↵	#↵	Starts the motion profile with number 1 on motor ‘A’.
!GPB↵	#2560↵	Queries the actual position of motor ‘B’.

8.7 Reference table: status and error messages

Acknowledgement codes (ackcode)	
Code	**Description**
‘’	Okay, no errors.
‘E50’	Command given cannot be performed in **WAIT FOR DISABLE** state.
‘E51’	Command given cannot be performed in **DISABLE** state.
‘E52’	Command given cannot be performed in **INIT** state.
‘E53’	Command given cannot be performed in **ERROR** state.
‘E54’	Command given cannot be performed in **RUN** state.
‘E55’	Internal error.
‘E56’	Wrong servo controller or motor selected. This error message is sent if the servo controller is in the RS232 mode and a command addresses a motor or servo controller that is not allowed in this mode. In the RS232 mode only the servo controller ’1’ and motors ‘A’ to ‘D’ may be used.
‘E57’	The selected motor is not present or ‘no drive’ has been selected as ‘drive type’.
‘E58’	Unknown error.
‘E59’	Wrong command format.
‘E60’	Sign error.
‘E61’	An attempt has been made to set the **RUN**, **INIT**, **FREEZE** or **STOP** flag although these flags are set in the parameter **IO configuration**. When these are set, the state of the flags is determined solely by the digital inputs of the servo controller.
‘E62’	The motion profile selected is not present in the servo controller.
‘E63’	An attempt has been made to start a motion profile which was not made for the actual motor type.
‘E64’	Value range exceeded.
‘E65’	Command too long.
‘E66’	The motor is not in the mode **Serial**. Select **Serial** in the parameter inspector under **\Drives\Drive X\Set Value Generation\Run Mode**.
‘E67’	This command cannot be used with slave motors.
‘E68’	This command cannot be used with the selected motor type.

State codes (statecode)	
Code	**Description**
‘W’	Servo controller in **WAIT FOR DISABLE** state.
‘R’	Servo controller in **RUN** state.
‘I’	Servo controller in **INIT** state.
‘D’	Servo controller in **DISABLE** state.
‘E’ + *syserr*	Servo controller in **ERROR** state. There is a system error. The coding of *syserr* is listed further below in the system error table.
‘E’ + *driveerr* + ‘A’	Servo controller in **ERROR** state. A fault has occurred on motor ‘A’. The coding of *driveerr* is listed further below in the motor fault table.
‘E’ + *driveerr* + ‘B’	Servo controller in **ERROR** state. A fault has occurred on motor ‘B’.
‘E’ + *driveerr* + ‘C’	Servo controller in **ERROR** state. A fault has occurred on motor ‘C’.
‘E’ + *driveerr* + ‘D’	Servo controller in **ERROR** state. A fault has occurred on motor ‘D’.

System error codes (syserr)	
Code	**Description**
'2'	Supply voltage too low for power part
'3'	Supply voltage too high for power part
'4'	Supply voltage too low for signal part
'5'	Supply voltage too high for signal part
'6'	Servo controller overheated or controller fault
'7'	Servo controller internal 12V missing

Motor fault codes (driveerr)	
Code	**Description**
'1'	Motor overloaded
'2'	Motor overheated
'3'	A following error has occurred.
'4'	The motor slider is missing.
'6'	An error has occurred during initialization.
'7'	Wrong motor type configured or motor defective.
'8'	No motion profile for the motor, or a motion profile selected is incompatible with the motor type.
'9'	Board over current. The motor has drawn too much current, possibly due to an wiring error or a short circuit on the phases.
'10'	Board over temperature detected. Possible causes: Motor draws too much current, insufficient controller cooling
'11'	AGND or 5VDC fuse blown. This can happen if a short circuit between motor phases and signal lines.

8.8 Reference table: position increment

The following table lists the values signaled back when the **PI** command is used. They may, however, alter with a future ASCII protocol version. Whenever possible therefore, the position increment should be queried with the **PI** command.

Motor type	Increment
LinMot®	19.53125µm
Stepper	1/8 Step
Solenoid	23.438mA

8.9 Reference table: speed increment

The table below lists the values signaled back when the **VI** command is given. They may, however, alter with a future ASCII protocol version. Whenever possible, therefore, the speed increment should be queried with the **VI** command.

Motor type	Increment
LinMot®	190.735µm/s
Stepper	0.081469 Steps/s
Solenoid	0

8.10 Reference table: acceleration increment

The table below lists the values signaled back when the **AI** command is given. They may, however, alter with a future ASCII protocol version. Whenever possible, therefore, the acceleration increment should be queried with the **AI** command.

Motor type	Increment
LinMot®	238.419mm/s^2
Stepper	47.6836 Steps/s^2
Magnet	-

8.11 Reference table: current increment

The table below lists the values signaled back when the **CI** command is given. They may, however, alter with a future ASCII protocol version. Whenever possible, therefore, the current increment should be queried with the **CI** command.

Servo controller	Increment
Ex00 / Ex000	23.438mA

8.12 Reference table: motor designator

The motor designator identifies a motor in a system solution in which the ASCII protocol is used under RS485.

Controller ID	Motor A	Motor B	Motor C	Motor D
1	A	B	C	D
2	E	F	G	H
3	I	J	K	L
4	M	N	O	P
5	Q	R	S	T
6	U	V	W	X

A. Compatibility with previous releases

To ensure transition without problems to Release 1.3 of *LinMot®* Talk, configuration and motion profile data generated with the old versions may be imported into Release 1.3 of *LinMot®* Talk. The data files of Release 1.0, 1.1 and 1.2 are supported. Conversion is necessary when importing 'old' configurations. This is performed automatically by the **Parameter Inspector**. To avoid problems, new configurations arrived at in this way should be verified carefully.

Servo controllers loaded with software release 1.0 or 1.1 cannot be operated with *LinMot®* Talk Release 1.3. It is, however, possible to install *LinMot®* Talk Release 1.0, 1.1, 1.2 and 1.3 simultaneously on a PC. With a configuration like this, all servo controllers may be operated. Nevertheless it is essential that only one version of *LinMot®* Talk at a time is started. All software releases are found on the *LinMot®* homepage under the WWW address **http://www.linmot.com**.

	imports configurations from				**configures servo controllers with**			
	R1.0	R1.1	R1.2	R1.3	R1.0	R1.1	R1.2	R1.3
LinMot® Talk R1.0	✘				✘			
LinMot® Talk R1.1	✘	✘				✘		
LinMot® Talk R1.2	✘	✘	✘				✘	
LinMot® Talk R1.3	✘	✘	✘	✘			✘	✘

Table 1-1: Compatibility of *LinMot®* Talk software

Note: The most secure way to transfer the configuration and curves from one firmware release to another is to save the configuration by using the same *LinMot®* Talk version as the controller's firmware is and import the configuration with a *LinMot®* Talk of the same version as the destination controller's firmware is.

Note: A newer hardware version or revision requires always a newer firmware version. Firmware cannot support newer hardware as a matter of principle. The newest hardware is always supported by the latest firmware releases of 1.1, 1.2 and 1.3.

B. Service / Error display

Error display on version 2 controllers

When an error occurs it is displayed by means of the blinking of at least one of the four LEDs on the front of the servo controller.

Ready LED

The Ready LED is ON as soon as the supply voltage for the processor is guaranteed and the system has started up correctly. Communication with the PC over the serial interface is only possible when this LED is ON.

The following table summarizes the blink codes and their meaning:

Fault LED	Stat A	Stat B	Description
● ~3Hz	off	off	HW system error: Hardware error in the servo controller.
2x● 1Hz			SW1 System error: Software error in the servo controller. The system software wasn't loaded successfully.
● ~1Hz			SW2 System error: Software error in the servo controller. The system software wasn't loaded successfully.
on	● ~2Hz	● ~2Hz	Generic fault: The exact error message may be displayed with the error inspector.
on		2x●	The supply voltage for the Power circuitry is too low.
		3x●	The supply voltage for the Power circuitry is too high.
		4x●	The supply voltage for the Signal circuitry is too low.
		5x●	The supply voltage for the Signal circuitry is too high.
	▬	6x●	The servo controller is too hot.
	1x● : Mot A 2x● : Mot B 3x● : Mot B 4x● : Mot D	1x●	The motor is overloaded.
		2x●	The motor is too hot.
		3x●	Following error.
		4x●	The slider is missing from the motor.
		6x●	The initialization was not completed successfully.
		7x●	Incorrect motor type configured or damaged motor.
		8x●	A referenced motion profile for a motor is missing. Selected motion profile not valid for actual drive type.

Table 2-1: LED Error Code Table for version 2 controllers

Legend			
on	LED is ON	●	LED blinks shortly (ca. ¼ s)
off	LED is OFF	▬	LED blinks longer (ca. 1½ s)

Error display on version 3 controllers

When an error occurs on a version 3 controller, the error code will be displayed on the 7 segment LED module on the front. The following table shows the possible error codes with a short description.

Error code	Description
E0001	Missing or invalid parameter tree
E0002	Missing or invalid application
E0003	Controller type not supported
E0004	MT command interface not available
E0005	Timer watchdog error
E0006	Trap class A error
E0007	Trap class B error
E0008	No master found for slave motor
E0009	No external sensor defined
E000A	External sonsor not allowed on channel D
E000B	The application software needs an MT controller
E000C	Noise dead band is not supported on this device revision (must be set ot 0mm)
E0010	DCLV power too low
E0011	DCLV power too high
E0012	DCLV signal too low
E0013	DCLV signal too high
E0014	Electronic fault
E0015	HW Error: internal 12V missing
E0101	Drive A: Too hot calculated
E0102	Drive A: Too hot sensor
E0103	Drive A: Following error
E0104	Drive A: Slider missing
E0106	Drive A: Init failed
E0107	Drive A: Drive type mismatch
E0108	Drive A: Curve error (wrong type oder number)
E0109	Drive A: Board over current
E010A	Drive A: Board over temperature
E010B	Drive A: AGND or 5VDC fuse blown
E0201	Drive B: Too hot calculated
E0202	Drive B: Too hot sensor
E0203	Drive B: Following error
E0204	Drive B: Slider missing
E0206	Drive B: Init failed
E0207	Drive B: Drive type mismatch
E0208	Drive B: Curve error (wrong type oder number)
E0209	Drive B: Board over current
E020A	Drive B: Board over temperature
E020B	Drive B: AGND or 5VDC fuse blown

Error code	Description
E0301	Drive C: Too hot calculated
E0302	Drive C: Too hot sensor
E0303	Drive C: Following error
E0304	Drive C: Slider missing
E0306	Drive C: Init failed
E0307	Drive C: Drive type mismatch
E0308	Drive C: Curve error (wrong type oder number)
E0309	Drive C: Board over current
E030A	Drive C: Board over temperature
E030B	Drive C: AGND or 5VDC fuse blown
E0401	Drive D: Too hot calculated
E0402	Drive D: Too hot sensor
E0403	Drive D: Following error
E0404	Drive D: Slider missing
E0406	Drive D: Init failed
E0407	Drive D: Drive type mismatch
E0408	Drive D: Curve error (wrong type oder number)
E0409	Drive D: Board over current
E040A	Drive D: Board over temperature
E040B	Drive D: AGND or 5VDC fuse blown
$FFEC	RAM error
E1001	RTS: State is too long
E1002	RTS: Controller version not supported
E1003	RTS: Wrong firmware
E1004	RTS: No script found
E1005	RTS: Illegal command
E8000	Profibus DP: Connection to master lost
E8001	Profibus DP: Address not valid
E8002	Profibus DP: Data out of range
E8003	Profibus DP: Invalid configuration from PLC
E8100	DN: Application needs DeviceNet controller
E8101	DN: DeviceNet MACID already in use
E8104	DN: Processor speed not supported by SW
E8108	DN: Unknown command
E8110	DN: Drive not specified in command
E8111	DN: Drive is not master
E8112	DN: Drive is not in serial mode
E8118	DN: Range error

Error code	Description
E8120	DN: Encoder does not exist
E8121	DN: Encoder is in SSI mode
E8130	DN: Curve does not exist
E8131	DN: Curve type mismatch Kurventyp
E8132	DN: Curve processing
E8140	DN: Unspecified CAN error
E8141	DN: CAN stuff error
E8142	DN: CAN form error
E8143	DN: CAN acknowledge error
E8144	DN: CAN bit1 error
E8145	DN: CAN bit0 error
E8146	DN: CAN CRC error
E8147	DN: CAN message lost
E8148	DN: CAN BOFF error
E8200	CO: Application needs CANopen controller
E8201	CO: Invalid address
E8202	CO: Data out of range
E8203	CO: Drive is not in serial mode
E8210	CO: Bus error
E8218	CO: Unspecified CAN error
E8219	CO: CAN stuff error
E821A	CO: CAN form error
E821B	CO: CAN acknowledge error
E821C	CO: CAN bit1 error
E821D	CO: CAN bit0 error
E821E	CO: CAN CRC error

Table 2-2: LED Error Code Table for version 3 controllers

C. Maintenance of servo controllers

The servo controllers have no parts requiring maintenance by the user. In normal operation it is not necessary to open these units. Care must be taken to ensure that the heat generated by them can be dissipated without problems. The heat sink should therefore be dusted off regularly and any other deposits cleaned away.

Fuses

The supply inputs of the servo controllers are fused against overcurrents. On the power PCB are two miniature fuses: one for the signal current and one for the power current. Their positions may be seen below.

Ex00-AT/MT/DP/DN

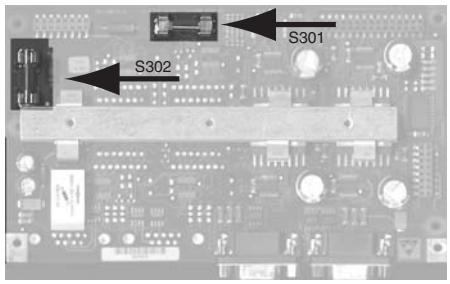


Figure C-1: *LinMot®* Ex00-AT/MT/DP fuses on power PCB

Fuse	Type
S301, motor supply	10A slow, ∅ 5mm x 20mm
S302, logic supply	0.5A slow, ∅ 5mm x 20mm

Ex000-AT/MT/DP/DN

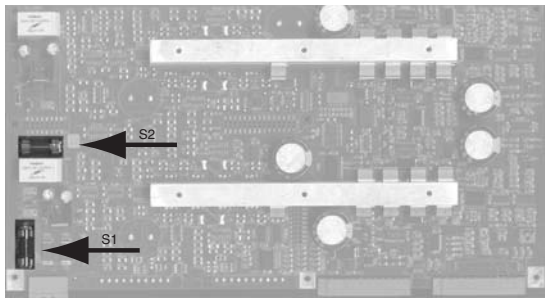


Figure C-2: *LinMot®* Ex000-AT/MT/DP fuses on power PCB

Fuse	Type
S1, motor supply	10A slow, ∅ 5mm x 20mm
S2, logic supply	0.5A slow, ∅ 5mm x 20mm

D. Maintenance of *LinMot®P* motors

The maintenance schedule below is based on a 5-day week with 8 working hours daily. Central European industrial operating conditions are assumed. Where conditions differ, as with severe and permanent fouling, direct sunshine, operation outdoors etc., the maintenance intervals must be shortened until empirical values for the particular application are obtained. Accordingly a distinction is drawn between the maintenance schedules for standard applications and first applications or arduous conditions.

Standard applications

This maintenance schedule should be used for standard applications

Time	Less than 120 strokes /min	120 to 360 strokes /min	over 360 strokes /min
commissioning	Inspection Lubrication	Inspection Lubrication	Inspection Lubrication
every 3 months	- -	Inspection	Inspection Lubrication
every 6 months	Inspection	Inspection Lubrication	Inspection Lubrication

First / arduous applications

This maintenance schedule should be used for first and arduous applications:

Time	Less than 120 strokes /min	120 to 360 strokes /min	over 360 strokes /min
commissioning	Inspection Lubrication	Inspection Lubrication	Inspection Lubrication
after first 8 hrs	Inspection	Inspection	Inspection
after first week	Inspection	Inspection	Inspection
all 3 months	Inspection	Inspection	Inspection Lubrication
all 6 months	Inspection Lubrication	Inspection Lubrication	Inspection Lubrication

Inspection

The following must be checked when inspecting the drives:

- Is the slider lubricated completely?
- Is the lubricant not ‘decomposed’?
- Can the slider be moved easily?
- Is the connector cable in good condition?

Cleaning

On no account may brushes or similar be used for cleaning purposes. No cleaning fluid containing solvents, kerosene or similar must be used.

1 Carefully withdraw the slider from the stator.

2 Clean the slider and stator with soft disposable paper, assisted by methylated spirit or alcohol.

3 Lubricate the slider and introduce it carefully.

Lubricating instructions

The lubricant reduces the friction between the chromium-nickel steel surface of the slider and the plastic plain bearing (POM or Delrin-based). In addition it prevents (fretting) corrosion. The lubricant employed must not attack the material of the plain bearing and must be temperature resistant up to 100°C. It must retain low viscosity at low temperatures and not evaporate.

The following table gives an overview for different lubricants.l

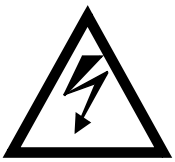
Application	Type	Distributor	Description
Plain bearings / food industry	*LinMot®* lubricant 0150-1950	*LinMot®*	Plain bearing paste for all *LinMot®* applications with long-time lubrication. USDA H1 approval. Synthetic low-temperature high-speed grease for rolling-contact and plain bearings, chains and seals.
	Klybersynth UH1 14-31	Klüber Lubrication	
Plain bearings	Molykote DX	Molykote	Alternative to *LinMot®* grease
Rolling-contact bearings	Microlube GBU Y 131	Klüber Lubrication	For rolling-contact and plain bearings, preferentially for high specific loads and influence of moisture and water.
Vacuum	Barrierta L55	Klüber Lubrication	High-temperature grease for rollers and ball bearings in conveying equipment and automatic baking ovens, also rolling-contact and plain bearings in electric motors.

Table D-1: Lubricants

E. Mechanical installation servo controllers

The *LinMot®* servo controllers can be panel mounted with two M5 screws each. The fixing plates are designed to facilitate mounting and dismounting the servo controllers.

The *LinMot®* servo controllers should be mounted vertically if possible. This will assure better cooling. *LinMot®* servo controllers have temperature monitoring, which switches off the power electronics in the event of an overheated heat sink.



CAUTION: When mounting the servo controllers, be aware of the housing temperature may reach up to 60°C, the heat sink up to 70°C. Make sure that there is adequate heat dissipation at the location of installation.

As already stated in the section on the current supply, the electronics must be earthed reliably..

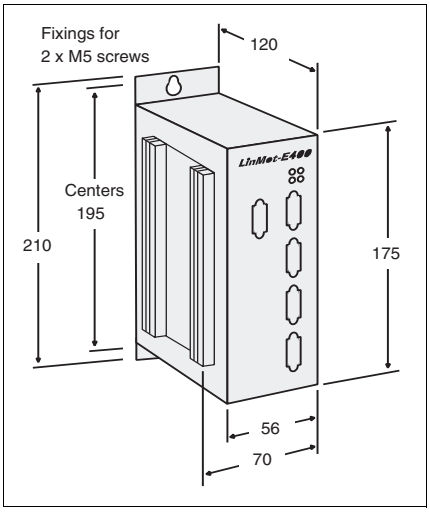


Figure 5-1: Dimensions of *LinMot® E100* series servo controller in [mm]

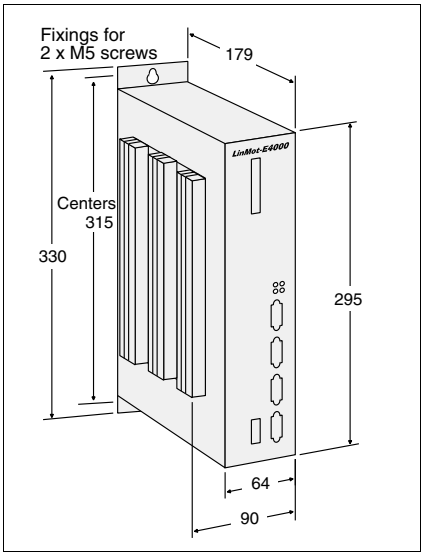


Figure E-2: Dimensions of *LinMot® E1000* series servo controller in [mm]

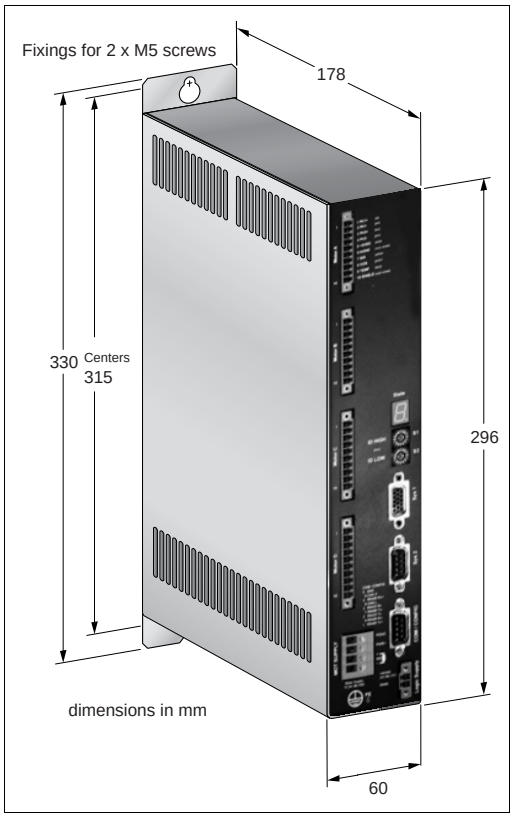


Figure E-3: Dimensions of *LinMot® E1001* series servo controller in [mm]

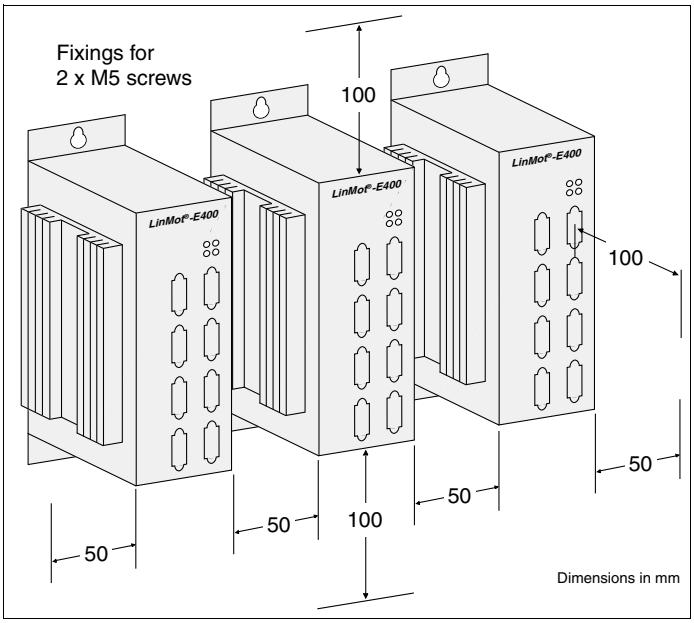


Figure E-4: Mounting of *LinMot® E100* series servo controllers in [mm]

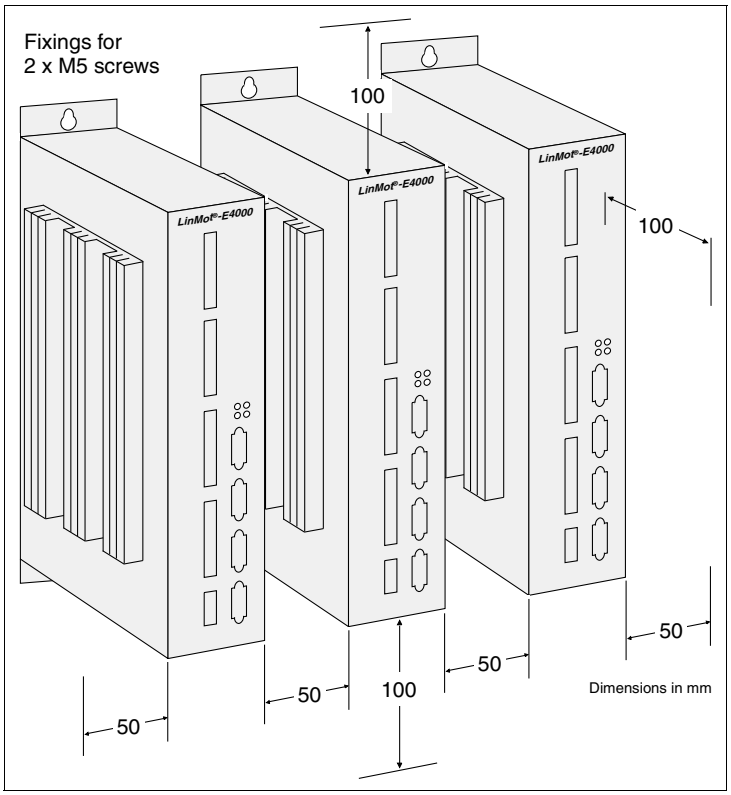


Figure E-5: Mounting of *LinMot® E1000* series servo controller in [mm]

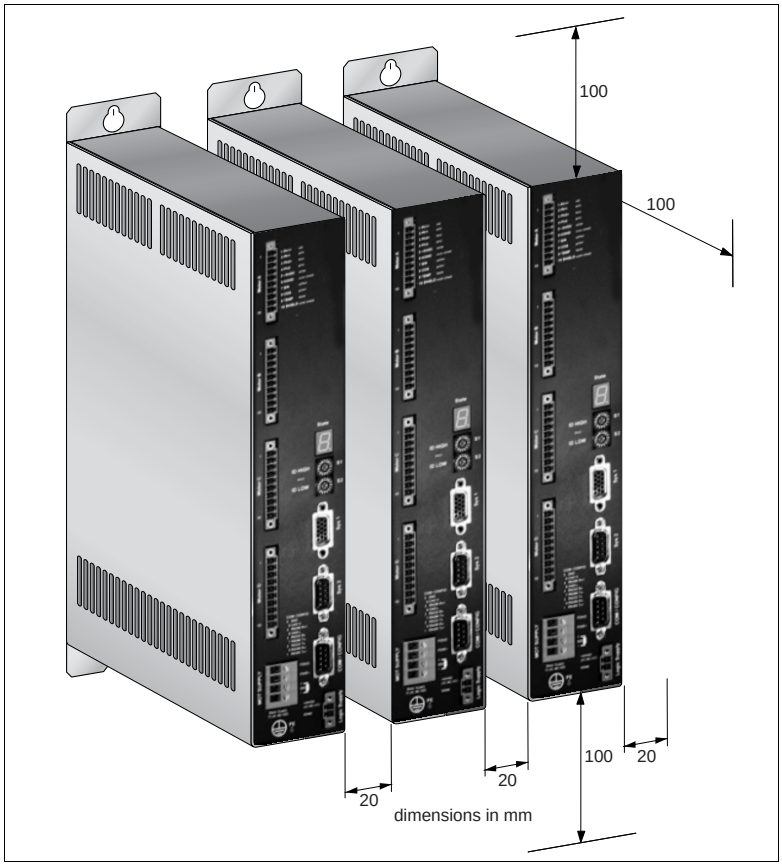


Figure E-6: Mounting of *LinMot® E1001* series servo controller in [mm]

F. Installation of the linear motors

The linear drives of the *LinMot® P* family feature a slide bearing between the moving slider and the fix stator. The requirements on this slide bearing construction are exceptionally high, due to the enormous dynamic properties and acceleration of the drives. For these reasons the following points must be observed.

Lateral forces

Due to the surface pressure caused by lateral forces excessive stressing on the slide bearing will result in reduced life of the linear motor. For this reason care should be taken in the application and installation in order to reduce lateral forces to a minimum.

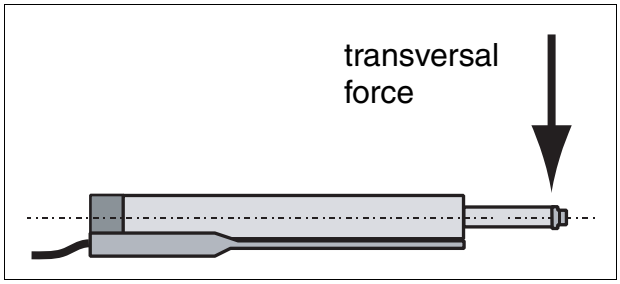


Figure 6-1: Lateral force

The linear drives of the *LinMot® P* series function best when operated as drive elements and are not used as guide or bearing elements.

Parallelism error

Binding of the system arises if the slider of the *LinMot®* P drives is used to move another longitudinally sliding machine part via direct coupling

In order to compensate for the parallelism errors, a flexible coupling (represented here by a bent line) must be used between the slider and the moving machine part.

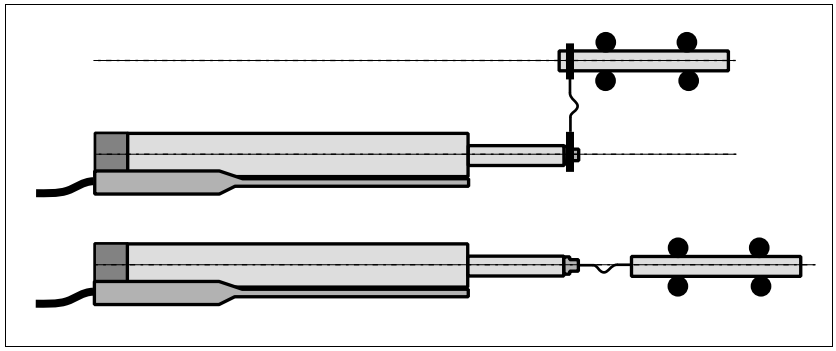


Figure 6-2: Compensation of parallelism errors

Horizontal mounting

If the linear motors are mounted horizontally, the key of the stator should be on the lower side (see figures above). If mounted like this, the slider mass is compensated by magnetic forces in the stator.

Mounting the stator

The stators are mounted by clamping. As clamping device the *LinMot®* flange should be used or a similar construction. Most important is a broad clamping surface in order to get a good heat dissipation. The clamping force should not be so tight as to compress the stator housing! (Hint: Don't use tightening tools with lever arm without an additional torque measurement).

Mounting of the load

The load mass must be mounted in a way, that only the end piece of the slider is held with the appropriate wrench (Caution: magnetic attraction). By no means should the sensitive slider tube or the slider end piece be clamped or used as a tightening tool.

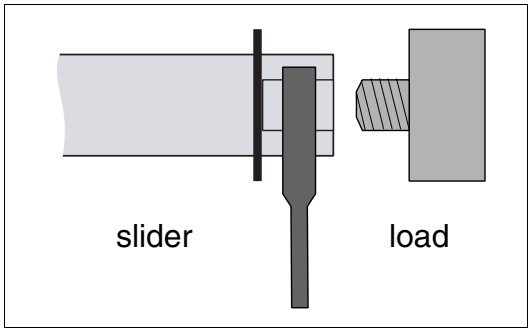


Figure 6-3: Load mass mounting

Handling the sliders

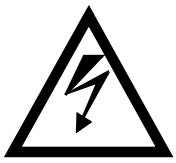
The sliders of the *LinMot® P* motors must be handled with great care. Even minor damage to the slider surface can cause a drastic reduction of its lifetime. The slider is a high precision machine element consisting of a thin steel tube and neodymium magnets. As it is designed for loads in the longitudinal direction, even just hitting an iron plate due to the magnetic attraction can cause permanent damage to the slider.

Entering the slider into the stator

Clean the slider with disposable paper first. Take special care that there are no metallic particles on the surface of the slider. Lubricate the slider in accordance with the instruction in section D.

Insert the slider with the notchless end into the stator.

Many helpful hints for mechanical solutions can be found in the *LinMot®* Design Manual (Art. No. 0150-2215)



CAUTION:

- Under no circumstances may a damaged slider be used further, as this can lead to permanent and non reparable damage to the stator!
- The slider of the *LinMot®* P motors contains neodymium magnets, that can cause damage to magnetical data medium or sensitive electronic devices by merely coming close to them.
- When manipulating the sliders, hitting them against iron parts, tools, etc. must be absolutely avoided, as this can lead to permanent damage of the slider (surface damage, bending). Further, hitting against other ferrous objects represents a danger of injury (Bruised fingers, etc.).
- The slider of the *LinMot®* P motors can reach temperature values that can cause burns if touched.
- The sliders of the *LinMot®* P motors are fast moving machine parts. The user is responsible for taking all the measures necessary to avoid any contact and the relative danger of injuries to living beings (cover, protection from contact etc.).
- Accumulations of dirt, in particular of ferrous chips (magnetic attraction!) or dry running of the slider can considerably shorten the slider's lifetime.
- The sliders may only be stored or transported in the special plastic case (with cardboard inlay) or already inserted and blocked in the stator.
- The sliders must be protected from dirt (particularly metal chips, etc.).
- Maximal storage temperature: 70 °C
- Both slider and stator must be cleaned and subsequently re-greased if they become dirty (see section D.).

G. Contact Addresses

SWITZERLAND

NTI AG
Haerdlistr. 15
CH-8957 Spreitenbach

Sales and Administration:	office@linmot.com
Tech. Support:	+41-(0)56-544 71 00 support@linmot.com
Fax:	+41-(0)56-419 91 92
Web:	http://www.linmot.com

USA

LinMot, Inc.
5750 Townline Road
Elkhorn, WI53121

Sales and Aministration:	877-546-3270 262-743-2555
Tech. Support:	877-804-0718 262-743-1284
Fax:	800-463-8708 262-723-6688
E-Mail:	us-sales@linmot.com
Web:	http://www.linmot-usa.com

- Please visit http://www.linmot.com/ to find the distributor near you.

Index

Digits

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

Z